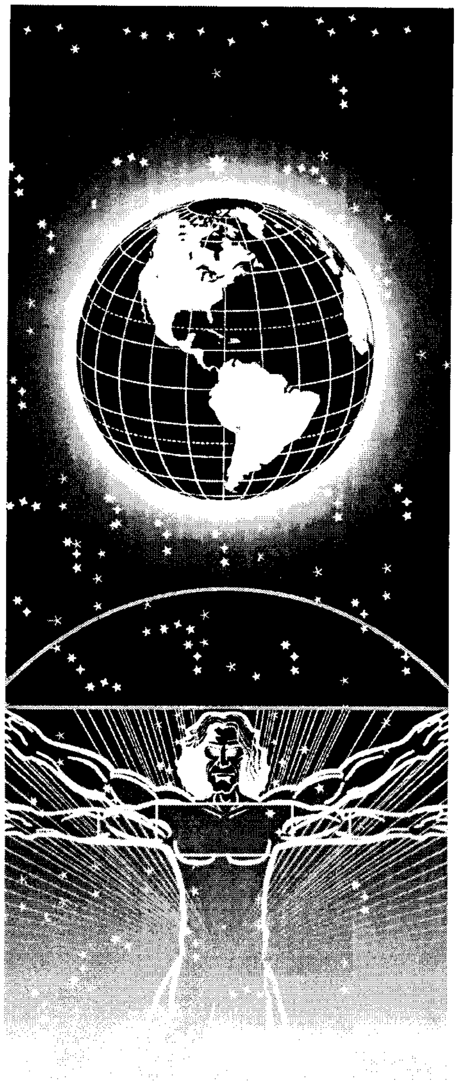
AFRL-HE-AZ-TR-1999-0207



UNITED STATES AIR FORCE
RESEARCH LABORATORY

INTELLIGENT TOOLS AND INSTRUCTIONAL SIMULATIONS

William R. Murray
Michelle Sams
Michael Belleville

Teknowledge Corporation
1810 Embarcadero Rd.
Palo Alto, CA 94303

August 2001

AIR FORCE RESEARCH LABORATORY
HUMAN EFFECTIVENESS DIRECTORATE
WARFIGHTER TRAINING RESEARCH DIVISION
6030 South Kent Street
Mesa AZ 85212-6061

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **01 AUG 2001**	2. REPORT TYPE **Final**	3. DATES COVERED **01-05-1996 to 01-09-1997**

4. TITLE AND SUBTITLE **Intelligent Tools and Instructional Simulations**	5a. CONTRACT NUMBER **N66001-05-D-8603**
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER **62202F**
6. AUTHOR(S) **William Murray; Michelle Sams; Michele Belleville**	5d. PROJECT NUMBER **1123**
	5e. TASK NUMBER **B2**
	5f. WORK UNIT NUMBER **1123B217**
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Teknowledge Corporation,1810 Embacadero Road,Palo Alto,CA,94303**	8. PERFORMING ORGANIZATION REPORT NUMBER **; AFRL-HE-AZ-TR-1999-0207**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **Air Force Research Laboratory/HE, Warfighter Training Research Division, 6030 South Kent Street, Mesa, AZ, 85212-6061**	10. SPONSOR/MONITOR’S ACRONYM(S) **AFRL; AFRL/HE**
	11. SPONSOR/MONITOR’S REPORT NUMBER(S) **AFRL-HE-AZ-TR-1999-0207**

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Air Force Research Laboratory Technical Monitor: Dr Joseph S. Mattoon, AFRL/HEA

14. ABSTRACT
This intelligent tools and instructional simulations project was an investigation into the utility of a knowledge-based performance support system to support learning and on-task performance for using desktop applications. The Desktop Associate is a multiagent system that couples a pedagogical agent with a monitoring agent to help users with desktop applications such as spreadsheet programs, word processors, and Internet browsers. It is intented to carry out common business tasks such as evaluating loans, creating a corporate newsletter, or downloading Internet software. The Desktop Associate implementation described in this report is a proof-of-concept design and development effort. It has not been formally tested but encouraging results were received from an early evaluation of the design, development tasks, and its potential capabilities.

15. SUBJECT TERMS
Computer-based instruction; Computer training; Desktop applications; Desktop Associate; Distributed cognition; Instructional simulations; Intelligent tools; Intelligent tutoring systems; Knowledge-based performance support system; Knowledge-based systems

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Public Release**	**54**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

NOTICES

JOSEPH S. MATTOON
Project Scientist

DEE H. ANDREWS
Technical Director

JERALD L. STRAW, Colonel, USAF
Chief, Warfighter Training Research Division

Direct requests for copies of this report to:

Defense Technical Information Center
8725 John J. Kingman Road, Suite 0944
Ft. Belvoir, Virginia 22060-6218
http://stinet.dtic.mil

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)* August 2001	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* May 1996 – Sep 1997
4. TITLE AND SUBTITLE Intelligent Tools and Instructional Simulations		**5a. CONTRACT NUMBER** N66001-05-D-8603
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER 62202F
6. AUTHOR(S) William R. Murray Michelle Sams Michele Belleville		**5d. PROJECT NUMBER** 1123
		5e. TASK NUMBER B2
		5f. WORK UNIT NUMBER 17
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Teknowledge Corporation 1810 Embarcadero Road Palo Alto CA 94303		**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory Human Effectiveness Directorate Warfighter Training Research Division 6030 South Kent Street Mesa AZ 85212-6061		**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL; AFRL/HEA
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) AFRL-HE-AZ-TR-1999-0207

12. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
Air Force Research Laboratory Technical Monitor: Joseph S. Mattoon, AFRL/HEA

14. ABSTRACT
This intelligent tools and instructional simulations project was an investigation into the utility and design of a knowledge-based performance support system to support learning and on-task performance for using desktop applications. The Desktop Associate is a multiagent system that couples a pedagogical agent with a monitoring agent to help users with desktop applications such as spreadsheet programs, word processors, and Internet browsers. It is intended to carry out common business tasks such as evaluating loans, creating a corporate newsletter, or downloading Internet software. The Desktop Associate implementation described in this report is a proof-of-concept design and development effort. It has not been formally tested but encouraging results were received from an early evaluation of the design, development tasks, and its potential capabilities.

15. SUBJECT TERMS
Computer-based instruction; Computer training; Desktop applications; Desktop Associate; Distributed cognition; Instructional simulations; Intelligent tools; Intelligent tutoring systems; Knowledge-based performance support system; Knowledge-based systems;

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT UNCLASSIFIED	**b. ABSTRACT** UNCLASSIFIED	**c. THIS PAGE** UNCLASSIFIED	UNLIMITED	54	**19b. TELEPHONE NUMBER** *(include area code)*

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

CONTENTS

PREFACE

This effort is covered by Delivery Order No. 3 to Naval Research and Development (NraD) Contract N66001-95-D-8642 with Teknowledge Corporation. The effort was funded by funds from the Defense Agency for Research Projects Agency (DARPA) under the Computer-Aided Education and Training Initiative (CAETI) program.

The research effort was conducted under Work Unit 1123-B2-17, Intelligent Tools and Instructional Simulations (ITTS). The Technical Monitor at the Air Force Research Laboratory, Human Effectiveness Directorate, Warfighter Training Research Division (AFRL/HEA) (formerly the Armstrong Laboratory, Human Resources Directorate) was Dr Joseph S. Mattoon.

This is the final report for the Intelligent Tools and Instructional Simulations contract, which describes the work on the Desktop Associate project from May 1996 through September 1997. Publication of the final report was delayed due to the laboratory reorganization and personnel reassignments.

Additional information on this project is available on-line at http://www.teknowledge.com/Desktop_Associate/index.htm.

INTELLIGENT TOOLS AND INSTRUCTIONAL SIMULATIONS: DESKTOP ASSOCIATE

INTRODUCTION

The Desktop Associate project is an investigation into the utility and design of a knowledge-based performance support system to support learning and on-task performance for using desktop applications. The Desktop Associate is a multiagent system that couples a pedagogical agent with a monitoring agent to help users with desktop applications such as spreadsheet programs, word processors, and Internet browsers, and to carry out common business tasks, such as evaluating loans, creating a corporate newsletter, or downloading Internet software.

The target audience for the Desktop Associate (DA) is administrative users: secretaries, support staff, and other non-technical personnel that use desktop applications in their day-to-day work. DA provides just-in-time training: when a new task needs to be accomplished or when the user needs tutoring assistance, DA can be invoked from within the application. DA is intended to be used while performing work tasks to improve and expand a user's capabilities.

The Desktop Associate implementation described is a proof-of-concept design and development effort. DA has not been formally tested but we have received encouraging results from an early evaluation of our design, development tasks, and DA's potential capabilities.

MOTIVATION

Desktop applications have become so feature-laden and complex that most users are only familiar with a small set of capabilities. Typically they may only know 10 to 15% of an application's total functionality. Furthermore, they are often unaware of what they do not know. For example, in Word 6.0/95 a user trying to center a table in a page would not be able to do so without knowing about frames—these are intended to contain and layout text, pictures, tables, or drawings—and how to use them to perform page layout functions such as text wrapping, and centering the contained object. Instead, a user might try to select the table and use the text center alignment tool, which would just center the text in each table cell. Similarly, a new user may not realize that the "Same as Previous" button must be turned off—it is on by default—on the headers and footers toolbar if a new section is to have different numbering from the previous section. Training can improve the user's knowledge and skill for exploiting more of the capabilities of modern software applications. However, conventional training courses remove personnel from their job responsibilities and disrupts business operations. Also, users typically forget much of what they learn prior to being able to apply it on the job.

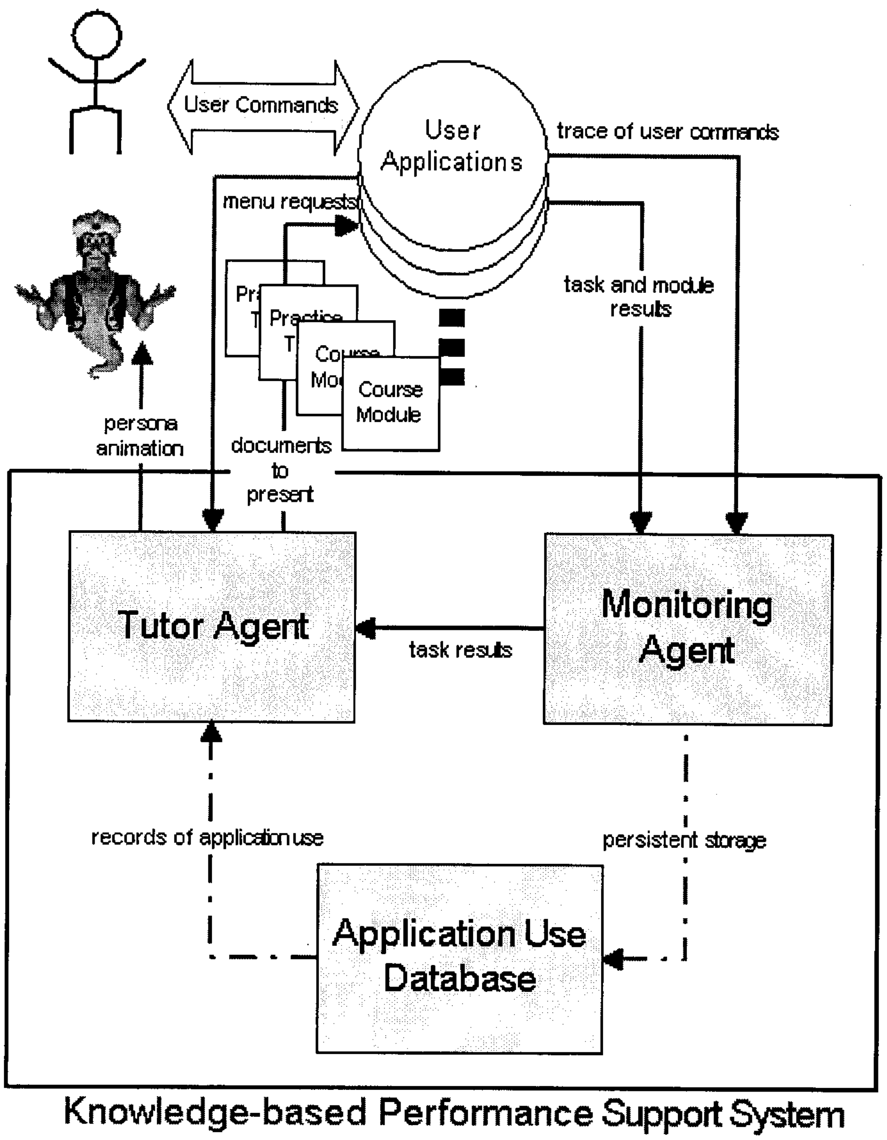


Figure 1. Overview of Desktop Associate System

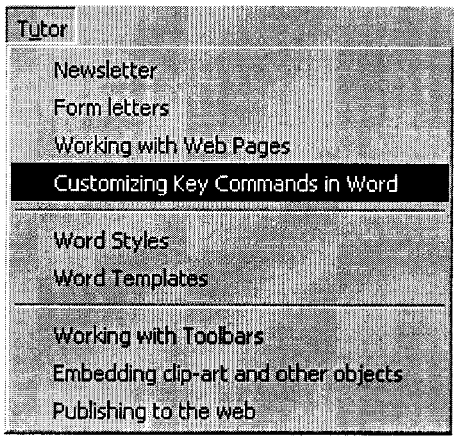
A second problem with conventional software training is that much of the current instruction for desktop applications is decontextualized: various application features will be explained (e.g., ranges in Excel) but the user will still need help in making the connection to day-to-day tasks that are often not provided. Instead, the instruction should be driven by the user's task needs and instruction should be provided in the application itself at the time it is needed.

It is this desire to provide instruction when needed and in the actual job environment that motivates the development of electronic performance support systems (Gery, 1991). One of the contributions of this work is that the multiagent system design of the Desktop Associate system architecture provides a general architecture for building knowledge-based performance support systems.

THEORETICAL FOUNDATIONS

In this section we look at the theoretical foundations behind the design of Desktop Associate based on research in performance support systems, knowledge-based tutoring, and student modeling.

Figure 2. Tutor menu for Word.



DISTRIBUTED COGNITION

When a person performs a task or solves a problem, it is often done with the aid of other resources. The knowledge brought to bear on the task can be said to be distributed among the individual and these other resources, such as, reference materials, computers, and other people. Perkins (1993) refers to this form of distributed cognition as "person-plus." In the case of desktop applications, the person possesses a certain body of knowledge from formal instruction, on-the-job training, and/or exploratory learning. When trying to solve a problem on the job, the person will refer to user manuals, on-line help, or solicit help from others. The degree of effectiveness of that person depends on their ability to find and effectively utilize the right resources. The goal of Desktop

Associate is to provide an integrated, intelligent and substantive resource, always available, that effectively enhances the user's distributed cognitive capability. DA is a user-regulated training and performance-support system, which could be called, "person-plus-associate."

COMPUTER-BASED INSTRUCTION (CBI), INTELLIGENT TUTORING SYSTEMS (ITS), AND STUDENT MODELING

Conventional computer-based instruction (CBI) can follow a number of approaches: fixed sequence, learner-controlled, adaptive-progression and automatic lesson branching. In fixed sequence systems, students progress through a prescribed set of instruction and exercises without deviation or adaptation. In learner-controlled systems, students are allowed to choose which topics to pursue. Adaptive-progression systems allow students who demonstrate proficiency to move forward to the lesson objectives more quickly (e.g., does not have to complete all the sample exercises). Automatic lesson branching is typically used to move students to remedial instruction and the move is based on a prespecified performance criterion (e.g., Atkinson, 1976; Goldstein, 1979; Kaplan & Holland, 1995). For example, students who achieve at least 80% on a multiple-choice test proceed to the next module, if not, then they are moved to a remedial module according to the system's assessment of their performance during practice or on-line testing.

Intelligent tutoring systems (ITS) typically build a student model and then tailor instruction based on that model. Some ITS systems have very precise and fine-grained models and employ tutoring strategies that adapt to any current state of the individual's changing level of performance, a process known as model tracing, such as in Anderson's ACT* and ACT-R tutors (Anderson, 1993). Recent research has shown, however, that the student model does not have to be exacting in detail to be useful and effective (e.g., Elsom-Cooke, 1989; Katz & Lesgold, 1992; McCalla, Greer, Barrie & Pospisil, 1992).

DA employs a combination of learner control, adaptive progression, and student modeling. DA is learner-controlled in that the student selects the topic and can choose whether to go through the instructional modules or practice tasks. DA employs a student model to determine whether instruction is needed and what type of instruction to recommend. The student model is not a fine-grained model tracer, but does build a historic model of the student's skill and knowledge. It retains that model, and if the same prerequisites (i.e., common underlying concepts) are required for other tasks or even other applications, it draws on that stored model and does not assess the student again in those areas. Its ability to do this rests in large part on the modeling of the domain knowledge that identified common underlying skills across tasks and applications. DA employs an adaptive progression approach in that, if a student demonstrates sufficient knowledge in a skill area, DA recommends that they go directly to some practice tasks. If they lack sufficient understanding of the skill area, then DA recommends specific prerequisite instructional modules.

Conventional training for desktop application skills typically occurs in a classroom context, with a platform instructor and multiple students. DA follows the principle of individually tailored instruction. Evidence supporting the power of tailoring instruction to the individual is ubiquitous in the literature. A great deal of research indicates that individualized instruction is far more effective than conventional group instruction. For example, Bloom, 1984, compared the achievement scores of individually tutored (IT) students, mastery learning students, and traditionally instructed students. Students in mastery learning are taught in the same way as traditionally instructed students but are assessed regularly to determine whether they "got it." If not, then remediation takes place. ML students performed one standard deviation (SD) above traditionally instructed students (or at the 84th percentile) and IT students performed two SDs above traditionally instructed students (or at the 98th percentile).

Effective human tutors follow a student modeling approach with adaptive instruction, and so do intelligent computer-based tutors. DA employs many of the following features.

Gamble and Page (1980) postulate that the following factors underlie effective human tutoring:

- work through problems selected by the student
- adjust to an individual student's background
- choose appropriate examples and problems for the student to work through
- cause the heuristics of the student to align with that of the tutor's
- measure student progress
- when required, review previously learned material

All automated instructional systems have been built with various subsets of these capabilities. In addition to these qualities, Regian and Shute (1992) note that, in theory, automated instructional systems should be able to surpass the instructional effectiveness of human tutors as they possess the additional capabilities:

- relentless in their persistence (being unable to "burn out"),
- can often graphically and behaviorally simulate desired learning contexts,
- can simulate psychomotor aspects of the transfer context (e.g., simulators and virtual world environments).

Research also shows that when target skills are complex, teaching component or prerequisite skills to proficiency can lead to faster learning of the complex skill (Stammers, 1980). If initial presentations are too complex, learners' attempts to practice may result in failure, anxiety, and eventual unwillingness to continue (Farber & Spence, 1953). In these cases, people learn more effectively if they first practice simpler parts of a task, before putting these parts together into the complete task. The approach is known as part-task training and is the basis of many training devices and simulators. DA takes an instructional approach that determines if students possess the necessay skills prior to tackling a task. If they lack the component knowledge and necessary skills, then DA presents modules on skill subsets that provide explicit instruction and sample exercises. Upon completing these modules, students can opt to complete practice tasks that integrate these skill subsets.

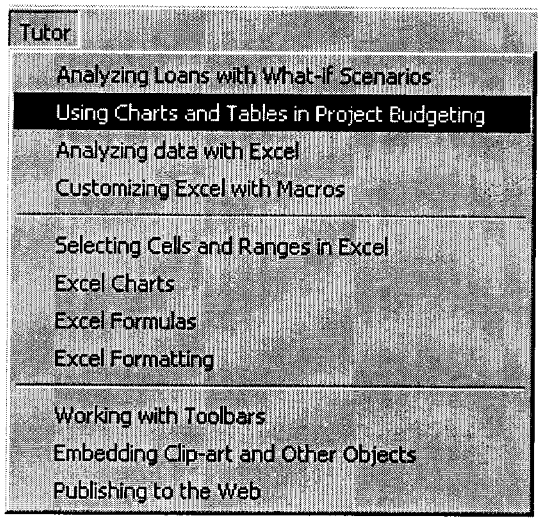

Tutor
Analyzing Loans with What-if Scenarios
Using Charts and Tables in Project Budgeting
Analyzing data with Excel
Customizing Excel with Macros
Selecting Cells and Ranges in Excel
Excel Charts
Excel Formulas
Excel Formatting
Working with Toolbars
Embedding Clip-art and Other Objects
Publishing to the Web

Figure 3. Tutor menu for Excel.

TRAINING TIME SAVINGS

It is anticipated that DA's use of tailored instruction and student modeling is instructionally effective and will also demonstrate a time savings. Many studies have demonstrated training time savings when using an intelligent tutoring system with student modeling. For example, subjects using the LISP tutor learned the knowledge and skills of LISP programming in one to two thirds the time it took a control group to learn the same material and attained the same or better criterion scores (in one study up to 43% better) (Anderson, Farrell, & Sauers, 1984). Students using the "Smithtown" tutor that teaches scientific inquiry skills in the context of microeconomics learned the material in half the time it took the classroom control group. Analysis also showed that the students learned the same material as the classroom students despite the fact that the tutor focused on teaching the inquiry skills not the subject matter (Shute & Glaser, 1991). Subjects using the "Pascal Tutor" learned equivalent knowledge and skills about Pascal programming in one third the time as learned through conventional instruction (Shute, 1991). The "Sherlock" tutor teaches avionics troubleshooting skills. After 20 hours of training, Sherlock subject skills were comparable to an advanced (expert) group of subjects and significantly better than the control group, who received on-the-job training (Lesgold, Lajoie, Bunzo, & Eggan, 1992).

OVERVIEW OF DA

Figure 1 shows an overview of DA and its relationship to the applications it supports. DA consists of two agents: a tutoring agent, and a monitoring agent. The monitoring agent is also called the Stealth Process since it, in essence, spies on other applications by watching the Windows messages they place on the processing queue. The tutoring agent provides instruction while the monitoring agent provides data collection. Data records showing application usage are stored in a database that the tutor agent accesses in constructing the user model. The user model is used to customize instruction. Finally, DA is represented on-screen by a persona, or graphical agent representation, using Microsoft's ActiveX agent. The applications themselves are used to present instruction and practice tasks. Document scripts analyze user solutions to check them. The monitoring agent reports back the results.

DA is designed to provide embedded training. For each application that DA supports, an additional Tutor menu is added to the standard toolbar. For example, Figure 2 shows the toolbar for Word, and Figure 3 shows the toolbar for Excel. In each case the menu is divided into three parts:

Business tasks—these menu items are tasks that can be done with the application. Selecting one of these lets a user practice the task, possibly after reviewing some prerequisite material.

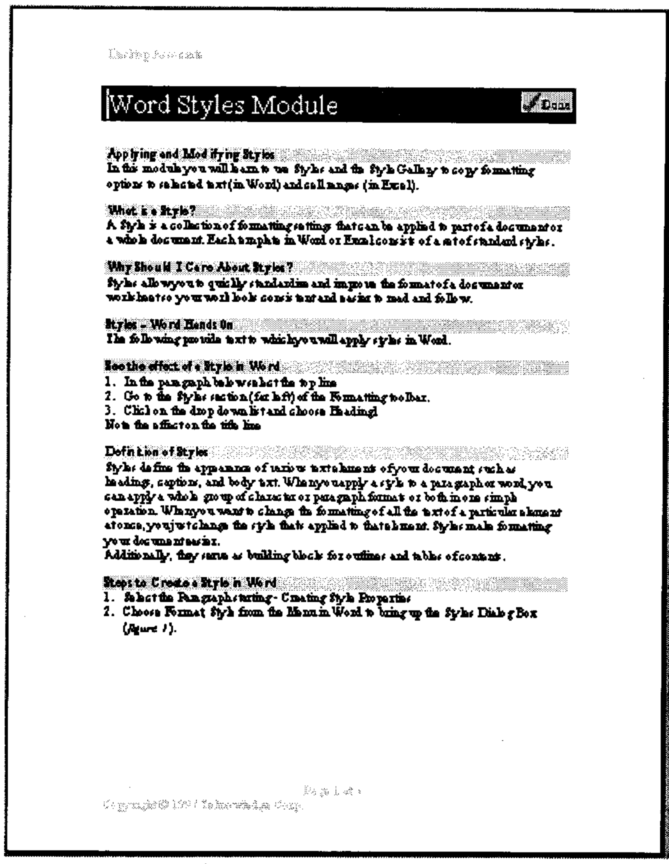
Application tools and features—these menu items are application-specific tools that can be used in many different kinds of tasks. For example, templates can be used to compose FAXes, memos, letters, and reports. These application concepts are introduced as prerequisite instructional modules in the context of business tasks if items from the first part of the menu are selected, but a user can also learn about them directly just by making selections here.

Cross-application functionality—the last set of menu items enables the user to learn about tools and features shared among many different applications. For example, once a user under-stands that objects such as how organization charts can be embedded in Word documents, the same knowledge can be applied to embedding charts in Excel or PowerPoint.

Because of these add-in menus,the Tutor part of DA is not invoked as a separate program but from *within* a particular application. That is, the user does not need to leave Word or Excel or any other DA application to use DA. In contrast, most tutoring programs would be separate standalone programs, outside of Word or Excel

The Stealth process runs invisibly in the background even when the tutor has not been explicitly invoked. That process monitors application and command use. It can be installed into a user's startup folder so it starts up automatically after rebooting the machine.

Instruction is delivered as documents that are inserted in the applications as they are used. For example, Figure 4 shows a Word module on Word styles, being presented from within Word itself. Sometimes another application may be used for presentation if it is more appropriate. So, an Excel module may be presented in Excel, but it also may be presented in Word, using embedded Excel spreadsheets for examples, when this is more illustrative.



Desktop Associate

Word Styles Module

Applying and Modifying Styles
In this module you will learn to use Styles and the Style Gallery to copy formatting options to selected text (in Word) and cell ranges (in Excel).

What is a Style?
A Style is a collection of formatting settings that can be applied to part of a document or a whole document. Each template in Word or Excel consists of a set of standard styles.

Why Should I Care About Styles?
Styles allow you to quickly standardize and improve the format of a document or worksheet so your work looks consistent and easier to read and follow.

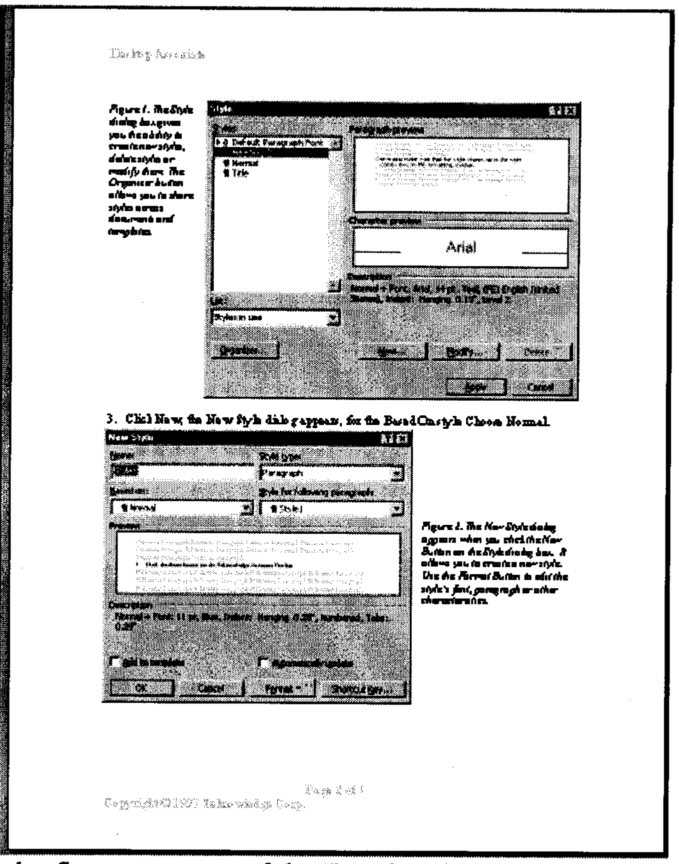
Styles – Word Hands On
The following provide text to which you will apply styles in Word.

See the effect of a Style in Word
1. In the paragraph below select the top line
2. Go to the Styles section (far left) of the Formatting toolbar.
3. Click on the drop down list and choose Heading1
Note the effect on the title line

Definition of Styles
Styles define the appearance of various text elements of your document such as headings, captions, and body text. When you apply a style to a paragraph or word, you can apply a whole group of character or paragraph formats or both in one simple operation. When you want to change the formatting of all the text of a particular element at once, you just change the style that is applied to that element. Styles make formatting your document easier.
Additionally, they serve as building blocks for outlines and tables of contents.

Steps to Create a Style in Word
1. Select the Paragraph starting - Creating Style Properties
2. Choose Format Style from the Menu in Word to bring up the Styles Dialog Box (*figure 1*).

Page 1 of 5
Copyright © 1997 Teknowledge Corp.



Desktop Associate

Figure 1. The Style dialog box gives you the ability to create a new style, delete a style or modify them. The Organizer button allows you to share styles across documents and templates.

3. Click New, the New Style dialog appears, for the Based On style Choose Normal

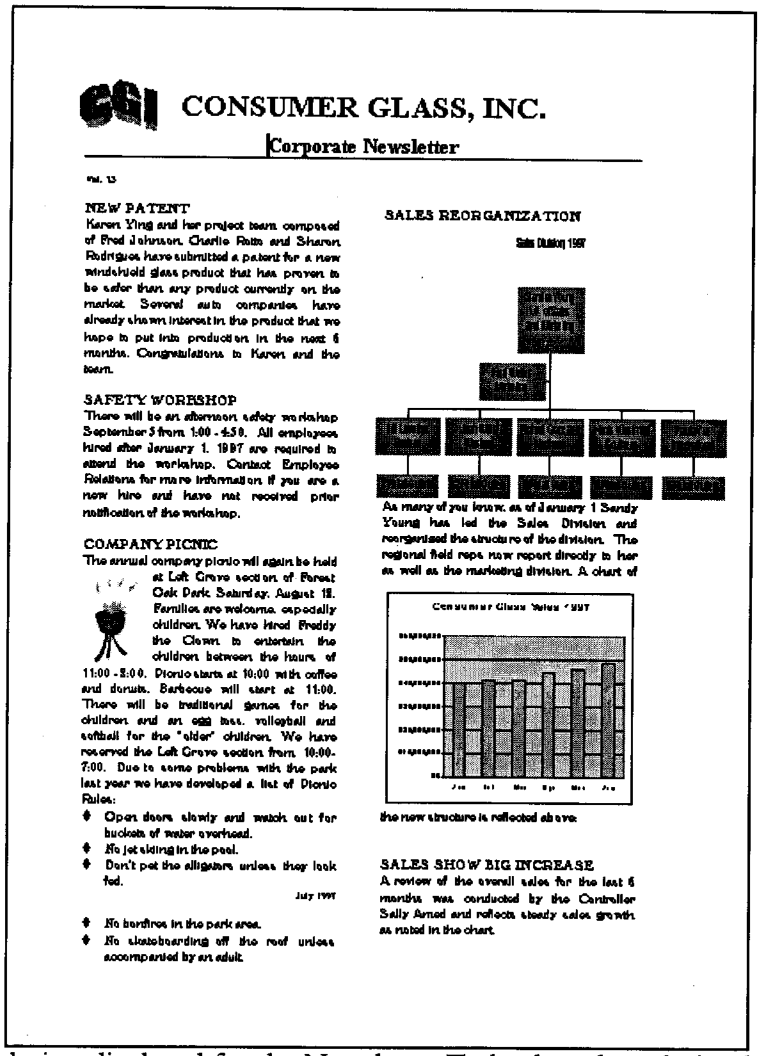
Figure 2. The New Style dialog appears when you click the New Button on the Style dialog box. It allows you to create a new style. Use the Format Button to edit the style's font, paragraph or other characteristics.

Page 2 of 5
Copyright © 1997 Teknowledge Corp.

Figure 4. Example of a course module: the first two pages of the Word Styles module

DA provides hints on demand when the user is performing a practice task. When a hint is requested the user's progress so far is analyzed and then a hint is selected. Similarly, when the user signals that they are done, the proposed solution is analyzed to see if the task has been performed correctly. The user can also look at the solution to a task for further assistance. For example, Figure 5 shows what the Newsletter Task looks like when successfully completed.

Note that DA does not attempt to infer what task a user is doing in the course of their day-to-day work and provide knowledge-based assistance for that task. Recognizing a user's goals and plans in such an unconstrained environment would be quite difficult and is beyond the scope of this contract.



CONSUMER GLASS, INC.

Corporate Newsletter

Vol. 13

NEW PATENT

Karen Ying and her project team composed of Fred Johnson, Charlie Ratto and Sharon Rodrigues have submitted a patent for a new windshield glass product that has proven to be safer than any product currently on the market. Several auto companies have already shown interest in the product that we hope to put into production in the next 6 months. Congratulations to Karen and the team.

SAFETY WORKSHOP

There will be an afternoon safety workshop September 5 from 1:00 - 4:30. All employees hired after January 1, 1997 are required to attend the workshop. Contact Employee Relations for more information if you are a new hire and have not received prior notification of the workshop.

COMPANY PICNIC

The annual company picnic will again be held at Loft Grove section of Forest Oak Park Saturday, August 12. Families are welcome, especially children. We have hired Freddy the Clown to entertain the children between the hours of 11:00 - 2:00. Picnic starts at 10:00 with coffee and donuts. Barbecue will start at 11:00. There will be traditional games for the children and an egg toss, volleyball and softball for the "older" children. We have reserved the Loft Grove section from 10:00-7:00. Due to some problems with the park last year we have developed a list of Picnic Rules:

- Open doors slowly and watch out for buckets of water overhead.
- No jet skiing in the pool.
- Don't pet the alligators unless they look fed.

July 1997

- No bonfires in the park area.
- No skateboarding off the roof unless accompanied by an adult.

SALES REORGANIZATION

Sales Division 1997

As many of you know, as of January 1 Sandy Young has led the Sales Division and reorganized the structure of the division. The regional field reps now report directly to her as well as the marketing division. A chart of the new structure is reflected above.

SALES SHOW BIG INCREASE

A review of the overall sales for the last 6 months was conducted by the Controller Sally Arned and reflects steady sales growth as noted in the chart.

Figure 5. Solution displayed for the Newsletter Task when the solution button is pushed.

RESEARCH CONTRIBUTIONS

The main research contributions of the Desktop Associate project are in the practical application of Bayesian user modeling, and DA's multiple-agent system design for knowledge-based performance support systems:

Student modeling—DA's approach to using Bayesian modeling for its student model provides for faster updating and simpler knowledge acquisition than earlier uses of Bayesian analysis (e.g., Tennyson & Rothen, 1979).

Knowledge-based performance support systems—DA provides an architecture for knowledge-based performance support systems where existing applications are used for both pedagogical presentation and for task assessment. It provides a general approach for adding knowledge-based performance support to applications that provide an API[1] along with scripting capabilities for hint generation and document analysis. The approach depends on having two interacting agents: one for tutoring and one for monitoring.

EXAMPLES

In this section we show two examples of how the Desktop Associate operates, as a user would see it. The first example shows DA being called from Excel, the second from Word. In this first example we just look at what DA does, in the second example we look at the student modeling and instructional planning that occurs behind the scenes.

LEARNING TO USE EXCEL TO CHART PROJECT BUDGETS

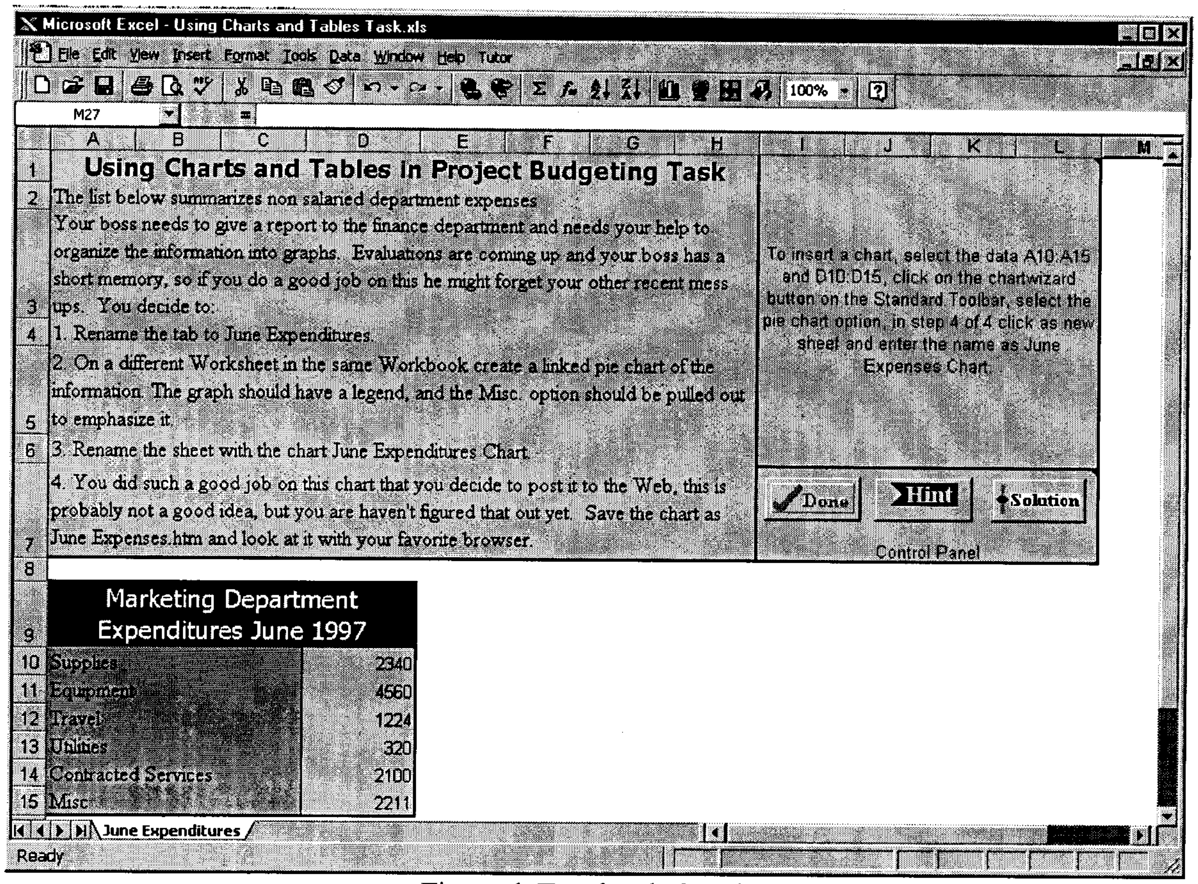
In this example we assume the user is a beginner in Excel and has not created Excel charts previously. The user would like to be able to chart their project expenditures and so they select the task 'Using Charts and Tables in Project Budgeting', the second menu item on Excel's tutor menu.

DA ASSESSMENT OF USER KNOWLEDGE

DA first assesses the user's knowledge of charting as this is a prerequisite skill for this task. This assessment would not happen if it had already occurred in the course of another task or if the user had already received DA instruction on charting. DA asks several questions, from easy questions to hard questions, to gauge the breadth and depth of the user's knowledge. Figure 6 shows an example of one of these questions. Since the user only answers the beginning questions correctly and one advanced question correctly while missing all the remaining intermediate-level, advanced-level, and expert-level questions, DA concludes that the user has a beginning level of knowledge of charting. (A later section on user modeling will discuss the Bayesian modeling approach used to make these inferences in more detail).
DA also uses application trace information to make its assessment. For example, if the user has used the chart wizard extensively and created a variety of charts, then it is more likely that the user is already proficient in charting. At present this capability is only partially implemented and much more work would need to be done to fully use the information collected by the monitoring agent. Currently, only a small number of questions can be answered automatically from examining the application data records.

DA recommends that the user learn more about charting before attempting the task. It always recommends review of prerequisite instructional material, unless the user has been assessed at an intermediate level of knowledge or better. It also never *requires* the user to go through the prerequisite material first. The DA approach is to always leave the user with final control, but to still provide guidance. The user can also choose to leave DA at this point or proceed directly to a practice task.

[1] API stands for application programmer's interface, a way of controlling a program from another application, without having to user the user interface.



Microsoft Excel - Using Charts and Tables Task.xls

Using Charts and Tables in Project Budgeting Task

The list below summarizes non salaried department expenses

Your boss needs to give a report to the finance department and needs your help to organize the information into graphs. Evaluations are coming up and your boss has a short memory, so if you do a good job on this he might forget your other recent mess ups. You decide to:

1. Rename the tab to June Expenditures.

2. On a different Worksheet in the same Workbook create a linked pie chart of the information. The graph should have a legend, and the Misc. option should be pulled out to emphasize it.

3. Rename the sheet with the chart June Expenditures Chart.

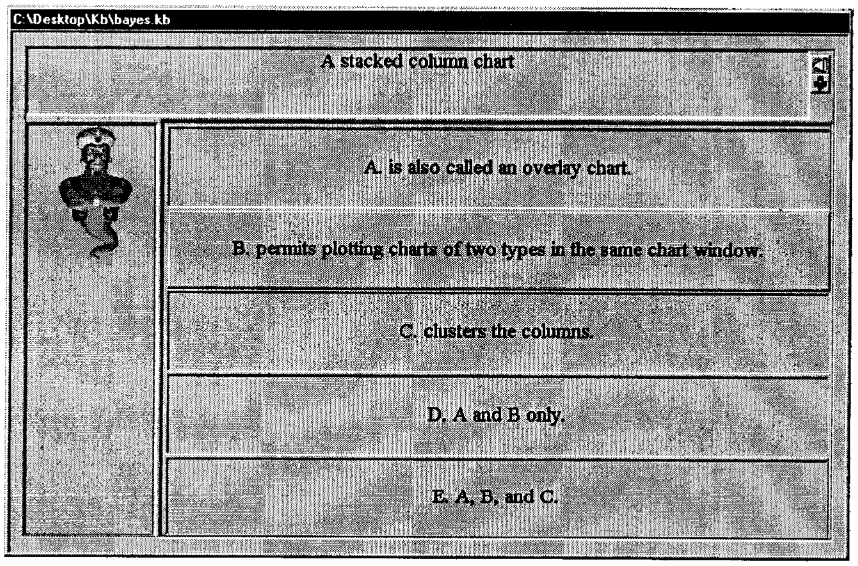
4. You did such a good job on this chart that you decide to post it to the Web, this is probably not a good idea, but you are haven't figured that out yet. Save the chart as June Expenses.htm and look at it with your favorite browser.

To insert a chart, select the data A10:A15 and D10:D15, click on the chartwizard button on the Standard Toolbar, select the pie chart option, in step 4 of 4 click as new sheet and enter the name as June Expenses Chart.

Marketing Department Expenditures June 1997	
Supplies	2340
Equipment	4560
Travel	1224
Utilities	320
Contracted Services	2100
Misc	2211

Figure 6. Excel task description.

Figure 7. Question assessing knowledge of Excel charts.



C:\Desktop\Kb\bayes.kb

A stacked column chart

A. is also called an overlay chart.

B. permits plotting charts of two types in the same chart window.

C. clusters the columns.

D. A and B only.

E. A, B, and C.

INSTRUCTIONAL MODULES

In our current scenario, the user chooses to accept DA's advice and completes the instructional module on charting. Most modules have small exercises for students to observe each instructional step. These sample exercises differ from the practice tasks in that they focus on a few features. The practice tasks integrate a number of features, even across applications, into a more complex job relevant task. DA does not at present provide assistance on these instructional exercises, as the initial focus was to provide assistance in the more difficult practice tasks. Ideally, DA would provide assistance on these instructional exercises as well.

In this case the user accepts DA's advice and reads through the module on charting. Most modules have small exercises to provide practice. The difference between the tasks and exercises at present is one of degree: the exercises are smaller and only test knowledge in one course module whereas the tasks are larger and can improve knowledge in multiple course modules, including multiple applications.

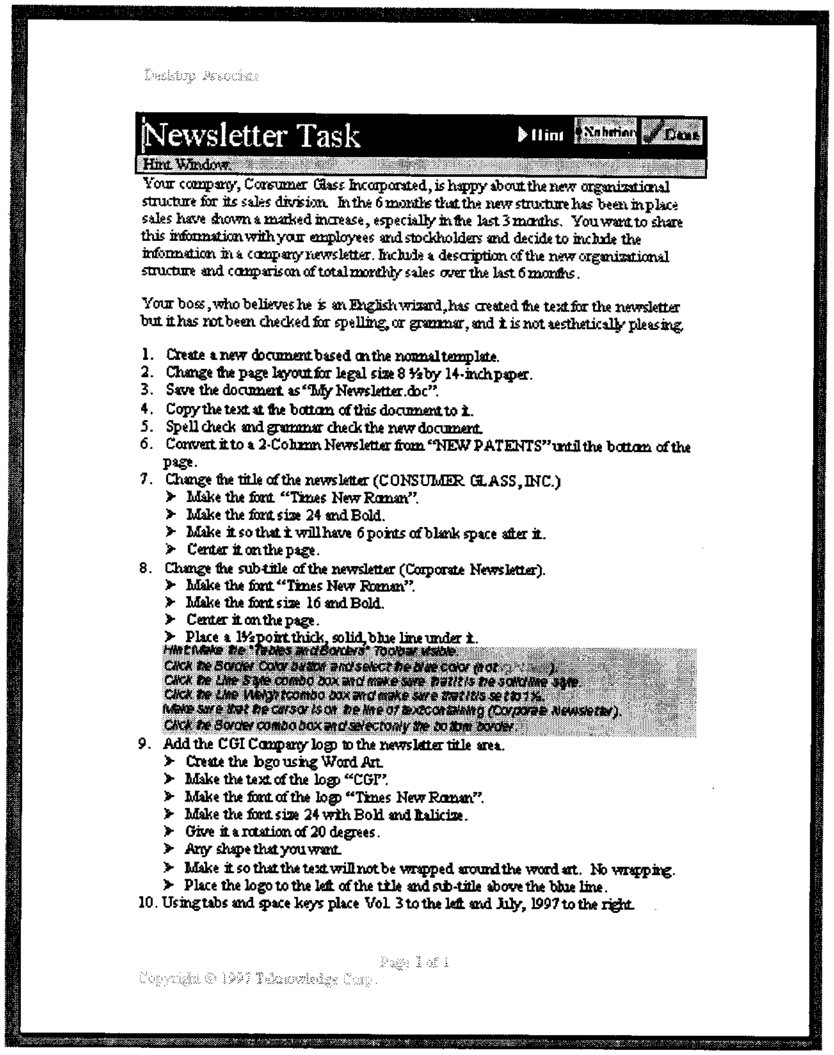
The user signals when they are through with the charting module by selecting the **Done** button. DA offers the user the previous choices, less the prerequisite review just completed. In our scenario, the user selects the task option.

PRACTICE TASKS

DA selects a task for the user within the task category specified, in this case, project budget charting. It requires that the user insert a chart on the same page and re-label the worksheet page. If the user does just one part of the task and not the other, either the **Hint** button or the **Done** button would prompt for completion of the remainder of the task. Figure 7 shows an example of a hint. When the user selects the **Done** button and the task analysis verifies the task has been completed, DA congratulates the user, suggests further practice on their examples of their own making, and returns the user back to the original application, in this case Excel.

LEARNING TO USE WORD TO CREATE A CORPORATE NEWSLETTER

Now let us look at a second example, examining the student modeling and instructional planning in more detail. This second example is in Word. The task selected by the student is to create a corporate newsletter.


Desktop Associate

Newsletter Task

Hint Window:

Your company, Consumer Glass Incorporated, is happy about the new organizational structure for its sales division. In the 6 months that the new structure has been in place sales have shown a marked increase, especially in the last 3 months. You want to share this information with your employees and stockholders and decide to include the information in a company newsletter. Include a description of the new organizational structure and comparison of total monthly sales over the last 6 months.

Your boss, who believes he is an English wizard, has created the text for the newsletter but it has not been checked for spelling, or grammar, and it is not aesthetically pleasing.

1. Create a new document based on the normal template.
2. Change the page layout for legal size 8 ½ by 14-inch paper.
3. Save the document as "My Newsletter.doc".
4. Copy the text at the bottom of this document to it.
5. Spell check and grammar check the new document.
6. Convert it to a 2-Column Newsletter from "NEW PATENTS" until the bottom of the page.
7. Change the title of the newsletter (CONSUMER GLASS, INC.)
   - Make the font "Times New Roman".
   - Make the font size 24 and Bold.
   - Make it so that it will have 6 points of blank space after it.
   - Center it on the page.
8. Change the sub-title of the newsletter (Corporate Newsletter).
   - Make the font "Times New Roman".
   - Make the font size 16 and Bold.
   - Center it on the page.
   - Place a 1½ point thick, solid, blue line under it.

   Hint: Make the "Tables and Borders" Toolbar visible.
   Click the Border Color button and select the blue color (not [illegible]).
   Click the Line Style combo box and make sure that it is the solid line style.
   Click the Line Weight combo box and make sure that it is set to 1½.
   Make sure that the cursor is on the line of text containing (Corporate Newsletter).
   Click the Border combo box and select only the bottom border.
9. Add the CGI Company logo to the newsletter title area.
   - Create the logo using Word Art.
   - Make the text of the logo "CGI".
   - Make the font of the logo "Times New Roman".
   - Make the font size 24 with Bold and Italicize.
   - Give it a rotation of 20 degrees.
   - Any shape that you want.
   - Make it so that the text will not be wrapped around the word art. No wrapping.
   - Place the logo to the left of the title and sub-title above the blue line.
10. Using tabs and space keys place Vol. 3 to the left and July, 1997 to the right.

Page 1 of 1

Copyright © 1997 Teknowledge Corp.

Figure 8. The Newsletter Task

This particular task also has not one, but two prerequisites: Word styles and Word templates[2]. The testing and prerequisite review is slightly longer, but the user can choose to either exit DA or go directly to the task whenever a module or task is completed. So they could just do the module on Word styles and then return to Word. The next day, if they reattempt the same task category they would not be assessed again over the same prerequisites as DA would remember the previous assessments. Instead, DA would recommend review of just Word Templates. The user could directly go to the Newsletter Task, as shown in Figure 8. The solution, shown earlier in Figure 5, is also available, along with hints, as before.

This time we will take a closer look at the behind-the-scenes activity: student modeling and instructional planning. First, the student's prerequisite knowledge is assessed. In this case, eight

[2] Word Art and Using Objects could also be considered prerequisites for this task, but are left out to simplify this demonstration.

questions are asked about Word styles and eight are asked about Word templates. As will be discussed later, skill level is divided into five categories (Novice, Beginner, Intermediate, Advanced, and Expert) and two questions are used in each category other than the lowest.[3] DA was designed to also supplement this information with the application data records.[4] Then DA could answer additional questions such as "Have you defined your own styles?" "Have you used any style-related tools, such as format-painter?" "Do you use templates other than the default template?" etc. from these records.

The student model is represented by a graph of skills where more general skills, such as Word skills, are at the top, and more specific skills, such as using the Format Painter toolbar button are at the bottom as shown in Figure 9. Part of the user model represents Word skills as shown in the Figure 9 diagram. The link from a parent to a child skill is a "Requires" link, i.e., to know the parent skill all the child skills must be known also.

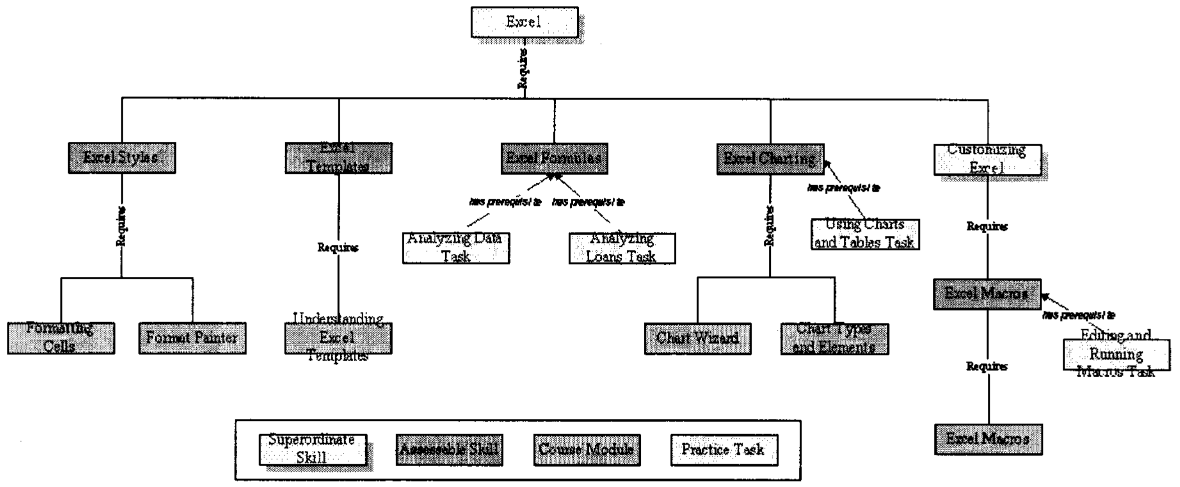


Figure 9. Part of the User Model under Word Skills.

Figure 9 shows the model used for Word skills. The current model breaks down knowledge of Word into knowledge of styles, formatting, templates, and customizing. The blue nodes marked as "assessable" indicate that there are banks of questions for these skills. Ideally all skills would be directly assessable, but currently only certain questions and course modules are available. The gray modules represent the course modules that were fully implemented. In some cases, such as for Excel formulas in the graph for Excel shown in Figure 10, a terminal node is blue or yellow, not gray,

[3] The user is placed in the lowest category (Novice) if they cannot answer any questions at all about the more difficult categories (Beginner, Intermediate, Advanced, and Expert).

[4] This part of DA is only partially implemented. These are records of which commands were used in Word, which were used in Excel, etc.

indicating that no course was completed for it. Ideally each terminal node would have a course module. Nonterminal nodes also do not have course modules associated with them as they represent more generic skills, such as knowledge of word templates in general, or knowledge or word styles in general. Such skills are broken down into their constituent skills where the instruction occurs. Instead, when a superordinate skill is encountered in the instructional plan a transitional statement, such as: "Now we are going to talk about Word templates. Templates allow you to create documents from ready-made copies, closer to what you want than if you had started from scratch. For example, you could have templates for project records or time cards."

The user model is represented as an overlay (Carr & Goldstein, 1977) to this skill breakdown. For example, if the user has no experience with either styles or templates, his assessment will place him in the lowest (novice) category for each skill, depicted in red in Figure 11.

Instructional planning also makes use of the same model shown in Figure 9. This planning sequences course modules for skills the user needs to develop to teach the particular skills missing. More sophisticated planning, not done at present, could take into account time and task difficulty when planning for user practice. Currently each prerequisite subtree is traversed and flattened into a linear sequence. The result is a set of plan fragments, one for each missing prerequisite. These fragments can be interpreted to deliver instruction as needed and would be tailored to each individual's knowledge / skill level and missing prerequisites. In this example two plan fragments are created: one to teach Word styles and one to teach Word templates. Figure 13 shows the resulting instructional plan.

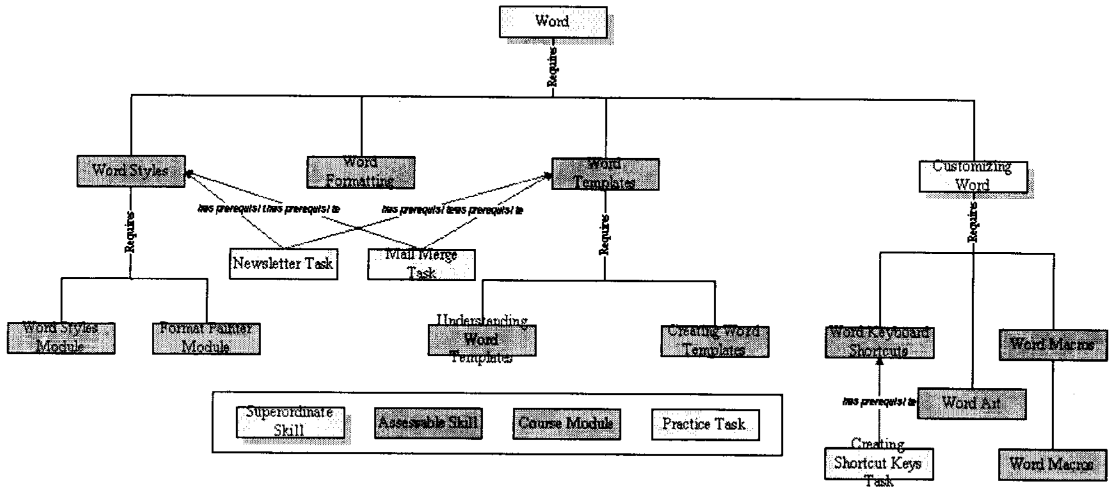


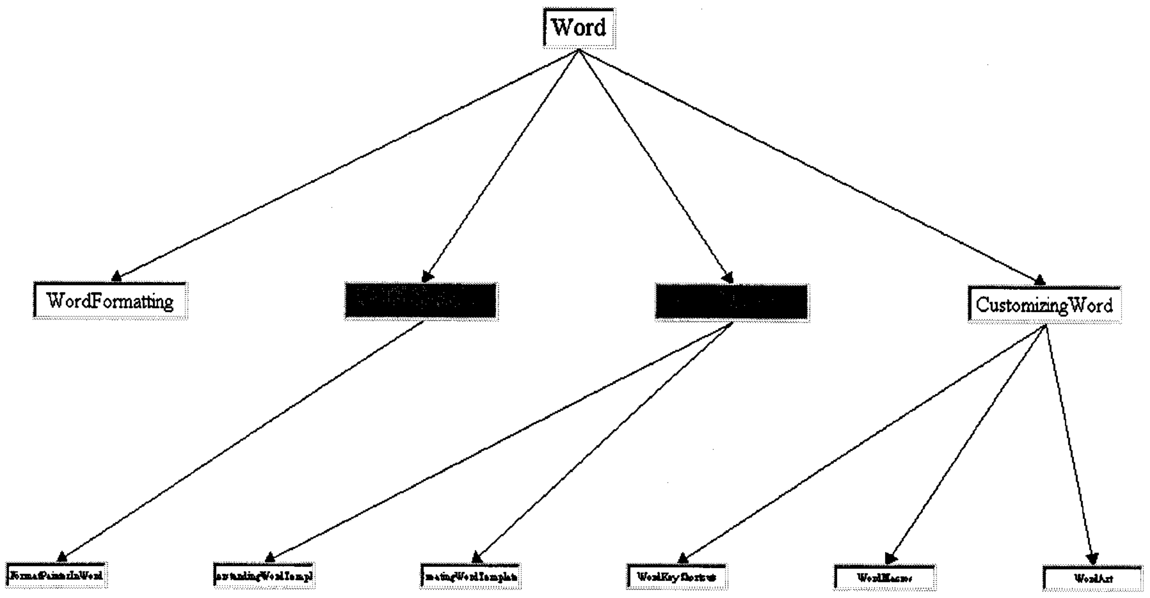
Figure 10. Part of the user model representing Excel skills

Note that a total order, one where the steps in one plan fragment precede the other, is not imposed. This partial ordering allows the user to choose which plan fragment (prerequisite to review) to pursue first. Also recall that the user can proceed directly to task instruction if desired.

ARCHITECTURE

This section discusses DA's architecture: how it is put together and how it operates behind the scenes. DA is a multiple-agent system consisting of two agents that communicate via a shared database. The first agent, the monitoring agent (also called the Stealth process or agent), monitors application use. The second, a tutoring agent, fields menu requests from the user to activate instruction. It builds and maintains a student model relying on information from both the Stealth agent and from the user's input. The Stealth agent describes the user's pattern of application and use of commands.

Figure 11. User model, after Word Styles and Word Templates are assessed

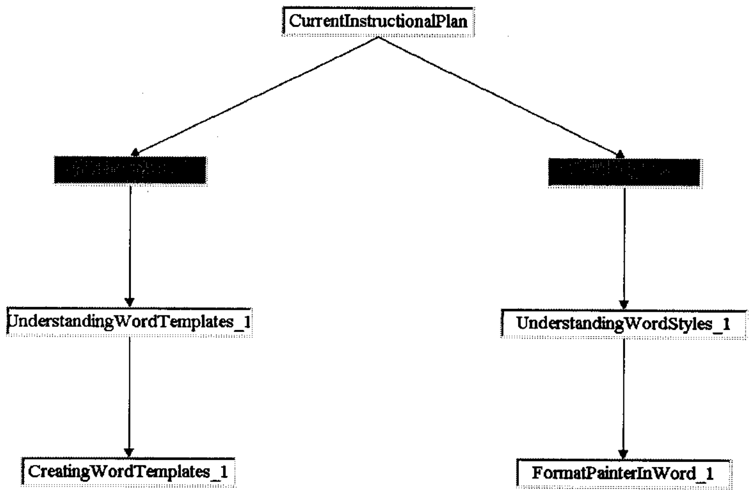


at the novice level.

THE MONITORING AGENT

The Stealth agent is placed in the user's startup folder and runs continually in the background. Its stores information collected on application use in an Access database that the Tutor agent can read. It also fields commands from the Tutor agent to display documents in the application. The documents themselves contain scripts to handle the hint, solution, and done buttons. The scripts send results back to the Stealth agent, which relays them to the Tutor agent.

Figure 12. The two instructional plan fragments generated after assessment.



MONITORING EXTERNAL APPLICATIONS

The Stealth agent is implemented with system level "hooks" and "controls" as described in Appendix II. External applications are monitored by a journal hook and associated DLL. The journal hook intercepts Windows commands being placed in the Windows message queue. These hooks allow it to intercept messages placed in the internal Windows message queue. The messages signal events such as menu selections, mouse movements, scroll commands, etc. Only relevant[5] messages are trapped and then an additional stage of translation is required. Messages only provide numeric parameters that denote commands, menu choices, toolbar buttons, etc. So, e.g., a message WM_COMMAND 17311 might be the format painter command in Word. But to translate that number into the command "Format Painter" requires another step. The Stealth application must next interrogate the application that sent the message to determine the actual application command, such as the Format Painter command, that caused the message. This translation could be performed in advance if we could guarantee that a user would not customize their toolbars or menus, but since they *can* change them (rearrange, add, and delete buttons), this dynamic lookup is required.

CONTROLLING EXTERNAL APPLICATIONS

The applications external to DA are controlled by OLE (Object Linking and Embedding) automation.[6] The Tutor agent needs to control the applications to use them for instruction. For example, it must be able to open documents within the application. It also controls the actions of its persona (the Genie animation) using OLE automation.

[5] Irrelevant messages are from other applications that DA does not provide tutoring for. For example, if the Calculator accessory were running its messages would be ignored.

[6] OLE is a means of supporting compound documents and data sharing.

USER MODELING

The user modeling relies on an approach to uncertain reasoning, called Bayesian Modeling. Essentially, the laws of probability are used as constraints to interpret evidence in light of known a priori probabilities and conditional probablities. Bayesian modeling can be computationally complex and the acquisition of the probabilities can be a time-consuming knowledge acquisition task. However, the method used here is straightforward and requires very few model parameters.

DA simplifies the modeling problem by only attempting to infer a single probability distribution—the probability that a student is at a certain skill level—based on evidence for or against each categorization. The evidence consists of multiple-choice questions grouped according to difficulty. This partitioning reduces the number of parameters required for the model. A further reduction results from relating these groups to the skill levels being tested. For example, if there are three skill levels (novice, learning, and experienced), then there would be two sets of question categories: one set of questions that only experienced users should be able to answer and a second set of questions that users in either the experienced or learning category should be able to answer. Users who have difficulty with both sets of questions are placed in the novice category.

The DA model actually uses five skill levels:

Novice level—the user has no exposure to the concept or skill, or erroneous ideas based on common or generic use of the terms or from other applications not relevant to the target application of inquiry.

Beginning level—the user can apply the skill in standard ways but has limited understanding of how it works or its general utility.

Intermediate level—the user can apply the skill in most of its common applications and has an accurate understanding of how the feature or skill works, and some understanding of how it interacts with other skills and features.

Advanced level—the user has extensive experience with the application and a thorough understanding of how it interacts with other skills.

Expert level—the user can apply the skill in original and non-standard ways, based on a deep mental model of the skill, its application, and how it interacts with other features or skills.

Thus four categories of questions are required: questions for beginners, questions for intermediate level users, questions for advanced users, and questions for expert level users. Again, users who cannot answer any of these groups are placed in the novice category.

Ideally users would be able to answer all questions at their skill level and at lower skill levels. In actual practice users deviate by making both slips and lucky guesses. In a slip they give a wrong answer to a question they should know. A lucky guess occurs when the user correctly answers a question they are not expected to know.

The DA model accounts incorporates these probabilities for slips and guesses, along with the a priori probabilities for the skill levels. It currently uses the following values:

probability of a slip = 0.1. This choice is somewhat arbitrary, we just expect this value to be close to 0.

probability of a lucky guess = 0.34. Most DA questions have four options so if a user just guessed randomly this probability would be 0.25. But we assume a higher probability as usually one answer can be ruled out in many questions even when you do not know the correct answer. In that case the guessing has a 1 in 3 chance of success, that is why the 0.34 probability is used here.

The a priori probabilities for the user population are based on the sample of users provided by Kelly. Most are expected to be beginners so the peak of the probability distribution is for that range then it falls off equally in both directions. But there are more higher-level categories so these are provided a probability, albeit half as large as either the novice or intermediate categories.

- a priori probability that student is a novice user = 0.2
- a priori probability that student is a beginner user = 0.4
- a priori probability that student is a intermediate user = 0.2
- a priori probability that student is a advanced user = 0.1
- a priori probability that student is a expert user = 0.1

Initially the model had a uniform distribution of 0.2 for each category but it was revised to provide a bias towards beginning users after the formative evaluation.

The significance of this approach is the savings in effort, both computationally and in terms of knowledge acquisition. For example, the ANDES physics tutor uses a Bayesian modeling approach that uses stochastic simulation to update the student model. The updating takes up to 40 seconds, which is acceptable for that tutor as the model is only used to select the next task. The approach used here requires time linear[7] to the amount of evidence, and provides subsecond responses.

The second savings is in knowledge acquisition. DA uses over 200 multiple-choice questions. Without its simplified approach of partitioning by question difficulty and using the same categories as skill levels, each question would require the specification of 5 probabilities as we need the probability that the question would be answered correctly for each skill level. So instead of two probabilities (the odds of a slip and a lucky guess) we would need 1000. In either case we would still need the a priori probabilities of the 5 skill levels. These can be obtained either from a subject matter expert's estimates or from measuring the distribution in an actual population sample.

[7] In contrast, general Bayesian belief updating takes exponential time, i.e., the amount of time required to update a network can grow exponentially in the size of the number of the nodes in the network [Russell and Norvig 95].

INSTRUCTIONAL PLANNING

DA carries out simple instructional planning in preparing its prerequisite reviews. A curriculum graph representing prerequisite relationships between skills is traversed to generate one or more modules to present for a concept. The resulting instructional plan fragments can be selected in any order by the user..

For example, consider the Newsletter task that has two prerequisites: a knowledge of Word styles and a knowledge of Word templates. Word templates has two subordinate skills: creating templates and using templates, as shown in Figure 12. DA traverses this three-node subtree of the curriculum graph to generate a high-level three-step plan:

1. Introduce the topic of word templates

2. Deliver the creating templates module

3. Deliver the using templates module

This instructional planning approach is similar to that used in Murray, 1990, in the Lower Hoist Tutor. That system, however, generated complete instructional plans that linearized (pieced together) all plan fragments. The current system does not impose a total order on all the plan fragments, as it is an associate system. Instead, unlike the earlier intelligent tutoring system, it lets the user choose which plan fragment (instruction path) to follow, or whether to skip all prerequisite review and proceed directly to task practice.

COURSEWARE

The courseware was written by a subject-matter expert (SME), for Word, PowerPoint, Excel, and Internet Explorer. Our SME designs and teaches courses on Word, Excel, and Visual Basic at the college level. He also provided the questions used in the student model evaluation that will be discussed later.

Each module presents an application feature or a feature shared across all the applications. Time and the scope of the research effort limited the number of modules that could be prepared; here is the final list of what is included in the current implementation:

- Word
    - Word styles
    - Format painter, used in Word
    - Word templates
    - Word macros
    - Customizing Word with shortcut keys
- Excel
    - Formatting cells
    - Chart types and elements

- Chart wizard
    - Excel macros
    - Excel templates
    - Format painter, used in Excel

- PowerPoint
    - Linked charts—Adding a chart from Excel linked to its data source using OLE
    - Presentation templates
    - Style checker
    - PowerPoint styles

- Internet Explorer
    - Searching the Internet
    - Downloading software
    - Saving text and links from web pages

- Shared functionality
    - Showing, hiding, moving, and docking toolbars
    - Customizing toolbar buttons: adding, removing, and reordering
    - Templates, in general
    - Word Art

Although almost every module has a module-specific task associated with it, there is also a set of multiskill tasks that are not specific to any single application concept. These multiskill tasks are:

- Analyzing data in Excel
- Analyzing loans in Excel
- Creating shortcut keys in Word
- Editing and running macros in Excel
- Embedding Word Art in Word documents
- Mail merge
- Newsletter
- Using charts and tables in Excel
- Internet searching
- Downloading software
- Saving links, text, and images

EMPIRICAL EVALUATION

INTRODUCTION

Details of Desktop Associate (DA) are provided elsewhere in this report, however, a brief recap of how one uses DA is provided here to refresh the reader's frame of reference. As the user selects each topic, DA presents a knowledge-based assessment instrument that determines whether the user has the prerequisite skills to attempt a practice task. If not, then DA selects the relevant instructional modules and recommends the user work through them. The instructional modules provide an overview of the feature and give explicitly detailed instructions that step the user through a sample exercise. After completing the instructional modules, DA suggests that the user try a practice task. The practice tasks are presented in a job relevant problem-solving context. Detailed steps on how to complete the task are available at the user's discretion via a hint button. Students are not allowed to proceed to the next step until the current step is successfully completed. A solution button reveals what the end product will look like and can be accessed at any time.

Based on resources available and the state of completion of DA, the evaluation study was scoped to achieve two main goals: 1) provide a test or external validation of the DA student model and 2) gain an indication of DA usability and effectiveness. A small representative user community participated in both parts of the study and only a sample of the software application was used. Following is a brief description of how the study was designed to achieve these two objectives.

PART I: MODEL EVALUATION

The DA student model is derived from a knowledge-based multiple-choice assessment instrument. The result of this assessment is a categorization of student skill level within each topic area. The skill level categories are novice, beginner, intermediate, advanced and expert. A measure of external validation was derived through a performance-based assessment session conducted by a subject-matter expert (SME).[8] The SME used in the study has extensive knowledge about the desktop domain and is a college-level instructor who is well practiced in assessing student skill level. Results of the student classifications from the knowledge-based DA and the performance-based session were compared to find the degree of classification congruence.

PART II: USER EVALUATION

DA usability and effectiveness was evaluated in two ways; 1) observations during a usability test and 2) subjective measures from a user evaluation survey. For the usability test, subjects worked through some DA instructional modules and practice tasks and were encouraged to verbally express their thinking processes. Observations were recorded by the study evaluator, who is a research

[8] The SME had the students define styles, use templates, use the format painter, etc. We deliberately asked him to focus on performance as he had provided many of the questions used in DA's assessment procedures.

psychologist with a specialty in human factors. Performance on the practice tasks, in particular, would be an indication of whether the instructional modules were effective in teaching the pre-requisite skills. Lack of learning would be indicated by the subjects attempting incorrect methods, a heavy reliance on the Hint key and Solution button, and asking the evaluator for help. Note that the evaluator did not give assistance directly related to skills that should have been learned in the instructional modules. The user evaluation survey was completed at the end of the session. The survey contained semantic differential and short answer items. Included in the survey were the topics of user interface, instructional design, whether anything new was learned, and the potential usefulness of DA on the job.

METHOD

SUBJECTS

Ten subjects participated in the study. Nine were from a temporary services agency and one was an employee of the host company site for the study. The host company employee was substituted when scheduled subject cancelled at the last minute. The subjects had varying degrees of familiarity with desktop applications and none had ever seen Desktop Associate. Of the ten subjects, eight were female and two were male.

PROCEDURE

Each subject participated in a single session for about 2 1/2 hours each.

Study Introduction

Each participant arrived at the host company site, signed in, and received a written copy of instructions for the study. Instructions contained a brief explanation of what the study was about and what to expect (i.e., the study procedures).

Part I: Model evaluation (1 hour)

<u>Performance-based assessment:</u> Each subject first interacted with the SME for 30 min. The SME asked each one to perform certain skills on the computer provided. Skills included: General Internet, Windows UI, Word Styles, Word Templates, Word Macros, Excel Styles, Excel Charts, General Views, Powerpoint Formatting, and Powerpoint Slide Layouts. Only skills which DA could assess were included.

<u>Knowledge-based assessment:</u> The subject then completed the on-line DA assessment survey covering the same skill categories. Subjects were asked to select the correct response from questions presented in a 4-alternative multiple-choice format. They were told if they did not know the answer to a given question, they were to make their best guess. (Note that the DA model factors guessing as part of its formula). The DA assessment for the selected skill categories took approximately 30 minutes to complete and typically covered 48 questions (8 questions per topic over 6 topics).

Part II: User evaluation (1 hour)

For the next hour, each subject participated in a hands-on tryout of DA. The study evaluator explained the intended use of DA in a typical office setting, told subjects that they would work through some examples, and were encouraged to make verbal comments during the session (likes, dislikes, confusions, etc.).

We asked subjects to select a specific initial task[9] under the Tutor menu on the top toolbar. They then completed the on-line DA assessment instrument that would ascertain whether they had the pre-requisite skills for the selected task. If the Bayesian analysis of the scores placed the user in the novice or beginner category, the DA genie would then recommend that they go through the pre-requisite instructional modules. Subjects then completed the relevant instructional modules with the illustrative exercises. Next they worked through the practice task, accessing the hint and solution buttons when they desired. Subjects were allowed to proceed at their own pace All subjects worked through the initial task, which was the Excel task 'Using charts and tables in Project Budgeting'. Some of the subjects completed additional tasks beyond this one.

RESULTS

A Likert scale was used to evaluate subject reactions. Asked if they liked DA overall, the average rating was 5.5 on a 1 to 7 scale; asked if they thought DA was instructionally effective, the average rating was 5.1 on the same 7-point scale. All subjects said they learned from DA. Details of both the survey and survey results appear in the Appendix.

The DA student model provided a +0.35 correlation with the subject matter expert, which was slightly better than the correlation between Kelly's own ratings of the subjects based on an internal CBT (computer-based training) assessment program used by Kelly; that correlation was +0.281. The DA correlation is lower than desired but we expect it would be improved if combined with application command use data gathered by the monitoring agent, and exercises added to the multiple-choice query of the user assessment. Additionally, the DA model matched the subject matter expert 74% of the time, was too conservative 22% of the time, and overestimated subject capabilities only 4% of the time. Both DA and the instructor used the same 5 categories in these ratings (Beginner, Novice, Intermediate, Advanced, and Expert). So these figures mean that DA and the SME picked the same category out of these five 74% of the time, DA picked a category lower than the SME's category 22% of the time, and DA picked a category higher than the SME's category 4% of the time. Thus, DA would offer instruction that would not be necessary 22% of the time, and fail to offer needed instruction only 4% of the time.

To compare DA to Kelly's measures the number of categories in DA was reduced from 5 to 3. DA's first two categories (Beginner and Novice) were mapped to Kelly's first category (Beginner). DA's next two categories (Intermediate and Advanced) were mapped to Kelly's second category (Intermediate). Finally, the highest categories in both were mapped together (DA Expert to Kelly Expert).

[9] The task was the most thoroughly tested and debugged task available which could be completed in the time allotted.

Note that the SME provided many of the questions for DA, but not all. Many were obtained from textbooks. More importantly, the SME instructed *not* to assess the subjects based on questioning, but only on performance, so we would be comparing DA's ability to assess based on the SME's questions to the SME's analysis of student performance and *not* the SME's questioning. Otherwise the primary difference would have been on-line versus human questioning.

INSTRUCTIONS TO SUBJECTS

The following shows the instructions to subjects. All subjects received these instructions and had several minutes to read over them upon arriving.

Desktop Associate Study

Welcome!

Here is a brief explanation of what this study is about and what to expect.

Teknowledge is doing some research on how to improve skills for desktop applications (MS Word, Excel, and PowerPoint). We have developed some prototype software, called Desktop Associate, and are having potential users, like you, try it out. The software is designed to assess your skill level on certain features, give you some instructional modules to learn from and some practice tasks. The software is still in the development stage and your frank comments are greatly appreciated.

** Everything you say and do here is confidential. We will not attribute any comments or results to any specific individual. We will roll up the results from many subjects and draw overall conclusions about our software. This is what you will do today:

Part I: (First Hour) Skill assessment

Our software has questions to find out how much you know about certain features, so it will present instruction on only those items that you need to learn or review. The purpose of Part I is to see whether our software questions will come up with the same conclusions as a human expert about what you know and don't know. The focus is on testing our assessment methods, not on how well you do. We asked for people with varying degrees of knowledge because this is the best way to see whether our methods are sensitive to these differences.

What you will do: A subject matter expert will interview you to determine your level of expertise with desktop applications. He will ask you some questions and have you perform some tasks on the computer. Then you will answer some questions on the computer.

Part II: (Second Hour) Trying out Desktop Associate

Our software is still in the development stage and we are very interested in how easy it is to use and how useful some of the features are. We want to know if the instructions are clear and whether it teaches what it is supposed to.

What you will do: An evaluator and developer will help you go through some of the instructional modules and exercises, and try out some of the practice tasks. Please feel free to ask questions and make comments. We will be taking notes on what to improve as you go through the tasks.

Part III: (Last half-hour) User Evaluation Survey

What you will do: We will give you a survey to rate the software and make some comments or suggestions. This will help us to develop a summary evaluation.

You are free to leave when you have completed the survey. Please turn it in to the receptionist.

We thank you for your time and effort. Have a great day!

LIKERT SURVEY FORM. The Likert survey form that subjects completed:

Desktop Associate
USER EVALUATION SURVEY

For the rating scales: Draw a circle around one of the dashes between the two opposite terms. Do not circle the term.

Instructional Modules

1. Organization of information
Confusing _ _ _ _ _ _ _ Clear

2. Instructional content
Difficult to understand _ _ _ _ _ _ _ Easy to understand

3. Did the instructional modules give you sufficient preparation to do the related tasks?
Insufficient preparation _ _ _ _ _ _ _ Sufficient preparation

Training Tasks

4.	Instructions	Confusing	_ _ _ _ _ _ _	Clear
5.	Sequence of Steps follow	Difficult to follow	_ _ _ _ _ _ _	Easy to
6.	Hints	Not helpful	_ _ _ _ _ _ _	Helpful
7.	Availability of Solution	Not much help	_ _ _ _ _ _ _	Helpful

Overall Reactions

8. Ease of use
Difficult _ _ _ _ _ _ _ Easy

9. Instructionally effective
Poor _ _ _ _ _ _ _ Good

10. Animated genie & spoken text
Negative reaction _ _ _ _ _ _ _ Positive reaction

11. Did you learn anything new? _____ yes _____ no

a) If yes, then what? ____

b) If no, then was it mainly because (choose just one)
_____ I already knew the skills
_____ The information was not clear
_____ There was insufficient practice time
_____ Other____

12. How useful do you think Desktop Associate would be as a
a) training program to help someone move up in skill levels.
not useful _ _ _ _ _ _ _ very useful

b) on-the-job aid to help someone figure out how to do something.

not useful _ _ _ _ _ _ _ very useful

13. In general, did you like DA? Did not like it _ _ _ _ _ _ _ Liked it a lot

Use Back Side Of Paper For Additional Comments

LIKERT SURVEY RESULTS. Results of the Likert survey follow:

User Evaluation Survey

Subject Means

Note: Means are in parentheses, based on a 7-point rating scale with 7 at the positive end of the semantic scale.

Instructional Modules

1.	(3.9)	Organization of information	Confusing - Clear
2.	(4.9)	Instructional content	Difficult to understand - Easy to understand

3. (5.0) Did the instructional modules give you sufficient preparation to do the related tasks?

Insufficient preparation - Sufficient preparation

Training Tasks

4.	(5.0)	Instructions	Confusing	- Clear
5.	(5.2)	Sequence of Steps	Difficult to follow	- Easy to follow
6.	(5.6)	Hints	Not helpful	- Helpful
7.	(5.1)	Availability of Solution	Not much help	- Helpful

Overall Reactions

9.	(4.8)	Ease of use	Difficult	- Easy
11.	(5.1)	Instructionally effective	Poor	- Good
12.	(4.5)	Animated genie & spoken text	Negative reaction	- Positive reaction

11. Did you learn anything new? 10 responded yes 0 responded no

12. How useful do you think Desktop Associate would be as a

a) (5.7) training program to help someone move up in skill levels.

not useful - very useful

b) (5.7) on-the-job aid to help someone figure out how to do something.

not useful - very useful

14. (5.5) In general, did you like DA?

Did not like it - Liked it a lot

RELATED WORK

MULTI-AGENT SYSTEMS FOR INFORMATION GATHERING

Sycara's work on Retsina (Sycara. et al., 1996) describes a collection of information-gathering and interface agents that work cooperatively to solve information-gathering tasks for users. Agents also have different roles in that architecture: there are interface agents, information agents, and task agents. The latter make domain-specific decisions, such as whether to buy or sell stocks, in the financial advisor application of Retsina. Matchmaker agents to link up agents with needs and agents that can satisfy those needs are also being investigated. All agents communicate with KQML[10].

In contrast, DA's multiple-agent architecture for knowledge-based performance support is much simpler: there only two agents, KQML is not used at present, and there are only two kinds of agents. What is similar is the use of different kinds of agents with different capabilities to provide a system that takes advantages of the complementary strengths of the different kinds of agents.

PERFORMANCE SUPPORT SYSTEMS

Most performance support systems are not knowledge-based. Typically they provide an integrated help, training, and task performance system. An agent approach to performance support systems is unusual, as is the approach of using a separate agent to specifically monitor user behavior with the system being supported.

INTELLIGENT TUTORING SYSTEMS THAT USE PEDAGOGICAL AGENTS

Intelligent tutoring systems that use pedagogical agents such as those of [Stone and Lester 96] provide an on-line persona and knowledge-based tutoring, but they are not intended to be performance support systems. Instead, they provide a separate instructional program. The intent of performance support systems is different: they are intended to provide instruction in the same environment as the application, using the application itself wherever possible (embedded training), rather than providing a simulation of the application.

LIGHTWEIGHT TUTORING AGENTS

Ritter's (Ritter & Koedinger, 1996) work on lightweight tutoring agents is similar in spirit to this work in that existing applications are used to present instruction through scripting and system events. It is different in that the tutoring agents are not intended to comprise a performance support system, rather they are intended to augment an intelligent tutoring system, and there is no autonomous monitoring agent that operates independent of the tutoring agents.

---

[10] KQML stands for knowledge query manipulation language. It is a high-level protocol for communicating requests, assertions, etc. between machine agents [Finon and Fritzon 94].

OFFICE ASSISTANTS

Finally, there are obvious visual similarities between the Microsoft Office Assistants in Office '97 and DA. Both have graphical agents and DA uses the Microsoft ActiveX Agent. The differences are:

- ➢ The Office Assistants perform a local Bayesian analysis[11] of the user's recent actions. There is no separate monitoring agent and no persistent storage of cumulative user actions. The analysis is used more to narrow the search for the top help items the user may wish to see, whereas in DA it is used to assess user understanding of task prerequisite knowledge.

- ➢ The Office Assistants are built into the Office applications whereas DA is an external program using program APIs and scripting capabilities to add tutoring functionality.
- ➢ DA is not restricted to Office applications. It was applied to Internet Explorer and it could be applied to Netscape, Visio, and other non-Microsoft applications.

- ➢ The Office Assistants can pop up unexpectedly; DA only appears when called. Sometimes the Office Assistants interpret the user's actions incorrectly and leap to offer assistance of the wrong kind or at the wrong time.

CONCLUSIONS

LESSONS LEARNED

What would we do differently? First we would focus on clarifying the Desktop Associate concept by proceeding to a prototype implementation prior to considering any issues of empirical evaluation. Only when there was software to be evaluated would we begin considering issues of evaluation. The early focus on evaluation provided a distraction from the hard work of creating a working implementation of Desktop Associate. Without a prototype to interact with, users could only speculate about what a desktop associate might be and how it might help them.

UTILITY OF THE STUDENT MODELING APPROACH

The Bayesian modeling approach used in DA is simple compared to other Bayesian modeling approaches that require complete updating of a Bayesian belief network. It provides the more principled approach of Bayesian analysis, based on probability theory, but is not hampered by the complexity of belief network propagation in a complex network with multiple levels and loops. The algorithm to compute it requires only a small number of parameters so the knowledge acquisition to acquire these parameters is much more simple than other Bayesian models that can require 100's of parameters (e.g., Collins, Greer, & Huang, 1996).

[11] The Office Assistants are part of the work on the "Lumiere Project: Bayesian Reasoning for Automated Assistance" at Microsoft, see http://research.microsoft.com/research/dtg/horvitz/lum.htm for more details.

ADVANTAGES OF EMBEDDED INSTRUCTION

DA shows that a knowledge-based performance support system can be embedded in existing applications provided that the architectures are sufficiently open: the applications provide an API, the operating system provides hooks to process messages, the applications are scriptable, and there are mechanisms for communicating between applications.

MULTIPLE-AGENT DESIGN FOR KNOWLEDGE-BASED PERFORMANCE SUPPORT

DA's dual agent approach of having a separate monitoring agent and a pedagogical agent for instruction makes sense when learning can occur outside of the instructional interventions and the monitoring agent can observe some evidence of this learning. Although other architectures for knowledge-based performance support can be designed, DA provides one specific example of how a performance support system can provide knowledge-based tutoring in particular.

CONTRIBUTIONS

The main research contributions of the Desktop Associate project are in student modeling, the use of autonomous agents to improve knowledge-based tutoring, and in knowledge-based performance support systems:

Student modeling—DA's approach to using Bayesian modeling for its student model provides for faster updating and simpler knowledge acquisition than earlier approaches.

Autonomous agents—DA's use of an autonomous agent to monitor application usage provides an ongoing way of monitoring user learning, even when DA tutoring is not being used. The additional data and its analysis can lead to improved knowledge-based tutoring for the monitored applications.

Knowledge-based performance support systems—DA provides an architecture for providing knowledge-based performance support where existing applications are used for both pedagogical presentation and for task assessment. It provides a general approach for adding knowledge-based performance support to applications that provide an API along with scripting capabilities for hint generation and document analysis.

REFERENCES

Anderson, J.R., Farrell, R., & Sauers, R. (1984). Learning to program in LISP. Cognitive Science, 8, 87-129.

Anderson, J.R. (1993). *Rules of the mind.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Atkinson, R.C. (1976). Adaptive instructional systems: Some attempts to optimize the learning process. In D. Klahr, (Ed.), *Cognition and instruction.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Barr, A; Beard, M.; & Atkinson, R.C. (1975) Information networks for CAI curriculums. In Lecareme, O.; and Lewis, R. (Eds.) Computers in Education, pp. 477 - 482. North-Holland, Amsterdam, The Netherlands.

Bloom, B. S, (1984). The 2-sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. Educational Researcher, 13(6), 4-16.

Carbonell, J.R. (1970). *Mixed-Initiative Man-Computer Instructional Dialogues.* Doctoral dissertation, MIT. Cambridge, MA.

Carr, B; & Goldstein, I.P. (1977) Overlays: A theory of modeling for computer-aided instruction. AI Lab Memo 406 (Logo Memo 45). Cambridge, MA: Massachusetts Institute of Technology.

[Clancey 87] Clancey, W. J. Knowledge-based Tutoring—The GUIDON Program. The MIT Press.

Collins, J.A.; Greer, J.E.; and Huang, S.H. (1996). Adaptive assessment using granularity hierarchies and Bayesian nets. Lecture Notes in Computer Science 1086. Proceedings of the Third International Conference, ITS '96. Frasson, Gauthier, and Lesgold (eds.), Springer, 569-577.

Elsom-Cook, M. (1989). *Guided discovery tutoring.* NATO Workshop on Guided Discovery Tutoring, Italy.

Farber, I.E., & Spence, K.W. (1953). Complex learning and conditioning as function of anxiety. Journal of Experimental Psychology, 45, 120-125.

Finin, T. & Fritzon, R. (1994). KQML as an agent communication language. The Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94), ACM. Press.

Gamble, A. & Page, C. V. (1980). The use of artificial intelligence techniques in computer-assisted instruction: An overview. International Journal of Man-Machine Studies, 12, 259-282.

Goldstein, I.P. (1979). *Principles of instructional design.* New York: Holt, Reinhart &Winston.

Gery, G. (1991). Electronic performance support systems. Ziff Institute.

Hayes-Roth, B. A (1985). Blackboard architecture for control. Artificial Intelligence Journal, 26, 251-321.

Hayes-Roth, B.; Garvey, A; Johnson, M.V.; & Hewett M. (1987). A modular and layered environment for reasoning about action (Technical Report KSL 86-38). KSL, Stanford.

Katz, S., & Lesgold, A. (1992, May). *Approaches to student modeling in Sherlock tutors.* Paper presented at the 3rd International Workshop on User Modeling, International Conference and Research Center for Computer Science (IBFI), Dagstuhl Castle, Germany.

Kaplan, J.D. & Holland, V.M. (1995). Application of learning principles to the design of a second language tutor. In V.M. Holland, J.D. Kaplan, & M.R. Sams (Eds.), *Intelligent language tutors: Theory shaping technology.* Mahwah, NJ: Lawrence Erlbaum Associates.

Lesgold, A., Lajoie, S. P., Bunzo, M., & Eggan, G. (1992). A coached practice environment for an electronics troubleshooting job. In J. Larkin, R. Chabey, & C. Cheftic (Eds), *Computer assisted instruction and intelligent tutoring systems: Establishing communication and collaboration* (pp. 201-238). Hillsdale, NJ: Lawrence Earlbaum Associates.

McCalla, G. I., Greer, J. E., Barrie, B. & Pospisil, P. (1992). *Granularity hierarchies. International Journal of Computers and Mathematics with Applications* (Special Issue on Semantic Networks).

Murray, W.R. (1990). A blackboard-based dynamic instructional planner. In Proceedings Eighth National Conference on Artificial Intelligence. pp. 434 - 441, 1990. AAAI Press / MIT Press.

Perkins, D. N. (1993). Person-plus: A distributed view of thinking and learning. In G. Solomon (Ed.), Distributed cognition (pp. 89-110), NY: Cambridge University Press.

Regian, J. W. & Shute, V. J., (1992). Automated instruction as an approach to individualization. In J.W. Regian and V.J. Shute (Eds.), *Cognitive approaches to automated instruction* (pp.1-13), Hillsdale, NJ: Lawrence Earlbaum Associates.

Ritter, S. & Koedinger, K. R. (1996). Towards lightweight tutoring agents. In Proceedings of the Seventh World Conference on Artificial Intelligence in Education (pp 91-98). Charlottesville, VA: Association for the Advancement of Computing in Education.

Shute, V. J. (1991). Who is likely to acquire programming skills? Journal of Educational Computing Research, 7(1), 1-24.

Shute, V. J., & Glaser, R., (1991). An intelligent tutoring system for exploring principles of economics. In R. E. Snow & D. Wiley (Eds.), Improving inquiry in social science: A volume in honor of Lee J. Cronbach. Hillsdale, NJ: Lawrence Earlbaum Associates.

Stammers, R.B. (1980). Part and whole practice for a tracking task: Effects of task variables and amount of practice. Perceptual and Motor Skills, 50, 203-210.

Stone, B. & Lester, J. (1996). Dynamically sequencing an animated pedagogical agent. Proceedings of the Thirteenth National Conference on Artificial Intelligence. pp. 424-431.

Sycara, K.; Pannu, A; Williamson, M; & Zeng, D. (1996, Dec). Distributed intelligent agents. IEEE Expert.

Tennyson, R.D. & Rothen, W. (1979). Management of computer-based instruction: Design of an adaptive control strategy. Journal of Computer-Based Instruction, 79(6), 812-818.

Wenger, E. (1987). Artificial intelligence and tutoring systems—Computational and cognitive approaches to the communication of knowledge. Morgan Kaufmann.

APPENDIX I: BAYESIAN STUDENT MODELING IN DESKTOP ASSOCIATE

HOW THE BAYESIAN MODELING APPROACH WORKS

The approach described here, used in the Desktop Associate knowledge-based performance support system, trades-off complexity of knowledge representation for a simpler linear-time algorithm for belief updating.

SKILLS AND PERFORMANCE DATA ARE REPRESENTED BY TREES

The model interprets a single skill with any number of performance data. The performance data can be actions that can be taken to solve some problem, questions to be answered, or any other data that can be classified as either being correct or not.

Figure 13 shows an example in the Desktop Associate. The skills in that system are knowledge of desktop application tools and features, such as the use of styles, templates, macros, and embedded objects in Word documents. The performance data are multiple choice questions.

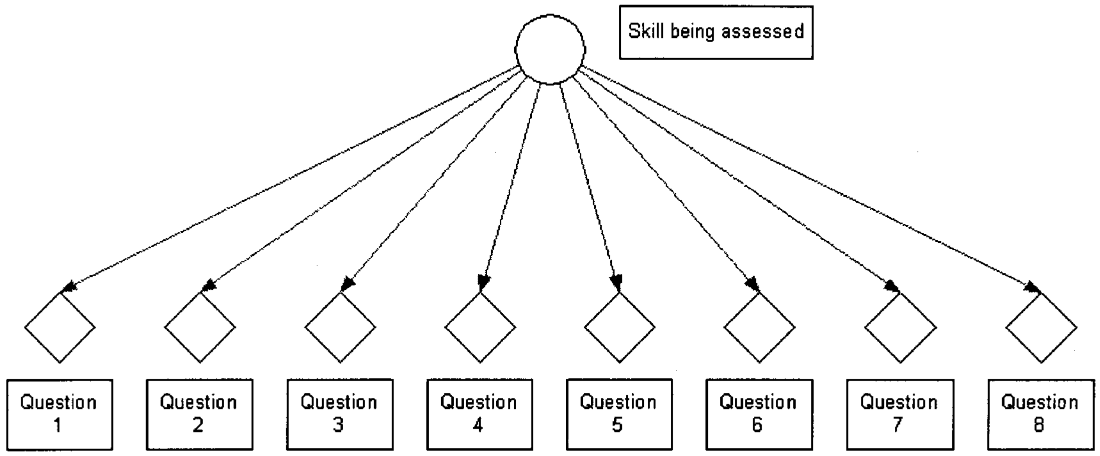


Figure 13. A skill being assessed using evidence from various questions.

The skill is represented by levels. For example, these could be beginner, intermediate, and advanced levels. In the Desktop Associate there are five levels:

Novice level—the user has no exposure to the concept or skill, or erroneous ideas based on English use of the terms or from other applications.

Beginning level—the user can apply the skill in standard ways but has limited understanding of why it works or its general utility.

Intermediate level—the user can apply the skill in most of its common applications and has a correct understanding of how the feature or skill works, and some understanding of how it interacts with other skills and features.

Advanced level—the user has extensive experience with the skill and a thorough understanding of how it intercts with other skills.

Expert level—the user can apply the skill in original and non-standard ways, based on a deep mental model of the skill, its application, and how it interacts with other features or skills.

Given that the skill has five levels then to apply Bayesian modeling we need the prior probabilities of each level along with the conditional probabilities for each question. For example, for question 3 we would need to supply the following conditional probabilities:

	Novice	Beginner	Intermediate	Advanced	Expert
Prob correct	P(correct | Novice)	P(correct | Beginner)	P(correct | Intermediate)	P(correct | Advanced)	P(correct | Expert)
Prob incorrect	P(incorrect | Novice)	P(incorrect | Beginner)	P(incorrect | Intermediate)	P(incorrect | Advanced)	P(incorrect | Expert)

Because the bottom row is 1 minus the top row, we really only need 5 probabilities for each question. For 8 questions that is 40 probabilities. In general if we have *n* skill levels and *q* questions for each we need nq probabilities plus the n prior probabilities of the skill levels. For *k* different skills, with the same skill levels, then we would need *knq* probabilities. As we can see even for a simple model the number of probabilities is large. For only 10 skills we already need 400 conditional probabilities. Even this model seems impractical.

Note that this model also assumes that the performance on each question is independent of every other question. If this were not the case then more complex joint probabilities would be required.

CONDITIONAL PROBABILITIES ARE DETERMINED BY DIFFICULTY LEVEL

The key to the simplified approach used in the Desktop Associate student model is to group the questions into categories of difficulty and tie these categories to the skill levels. By partitioning the questions into difficulty categories, each question in the same category has the same probability parameters as the other members. Now we need only provide *knc* probabilities where *c* is the number of categories. This simplification can provide a substantial improvement since the number of categories is typically much smaller than the number of questions for a skill. For example, we could have 3 categories (easy, moderate, and hard questions) and 10 questions in each category.

An additional simplification can be made: we can tie the question categories to the skill levels. For example, if we have 5 skill levels (e.g., novice, beginning, advanced, intermediate, and expert), we can have 4 corresponding question categories (e.g., beginning-level questions, advanced-level questions, intermediate-level questions, and expert-level questions). The reason we have *n+1* skill levels when we have *n* question categories is that we need an extra skill level for users whose skill is below the questions in any of the *n* question categories. In the Desktop Associate, the novice skill level corresponds to users who typically cannot even answer beginner-level questions.

The final reduction in probabilities required is a result of the transitive nature of these skill categories: if a user has reached a certain skill level that user should be able to answer all questions at that skill level *and all easier questions*. Similarly we expect them to miss all questions at any higher skill level. But in the real world users can make mistakes and sometimes miss questions they should know, or conversely be lucky and guess the right answer or perform the right action even though they do not have the knowledge that supports that choice. Without these slips and guesses the conditional probabilities would be either 1 or 0, depending on the user's skill level. For example, an expert level user would have probability of answering any question at the expert, advanced, or beginning level. Conversely, an intermediate-level user would have probability 0 of answering an advanced or expert question. But accounting for slips the expert's probability would be less than 1, by the probability of a slip, and the intermediate-level user's probability would be greater than 0, by the probability of a lucky guess.

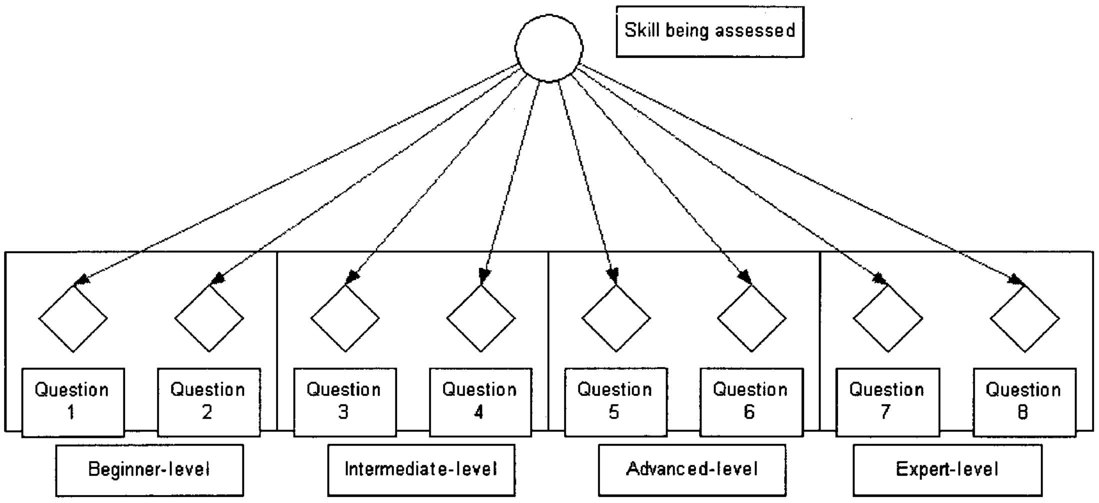


Figure 14. Partitioning questions into categories to reduce the number of model parameters

Call the probability of a slip *s* and the probability of a lucky guess *g*. *s* will be some number close to 1 such as 0.9 and *g* will be some number close to zero such as 0.1. If our performance data is multiple choice questions then we may be able to provide a better estimate for *g* as *1/m* where *m* is the number of choices in each question. In the Desktop Associate there are typically 5 questions so *g* is 0.20.

Now the probability that a beginning level question will be answered correctly is:

Skill Level	Novice	Beginner	Intermediate	Advanced	Expert
Prob correct	*g*	*1-s*	*1-s*	*1-s*	*1-s*
Prob incorrect	*1-g*	*s*	*s*	*s*	*s*

Now the probability that an intermediate level question will be answered correctly is:

Skill Level	Novice	Beginner	Intermediate	Advanced	Expert
Prob correct	*g*	*g*	*1-s*	*1-s*	*1-s*
Prob incorrect	*1-g*	*1-g*	*s*	*s*	*s*

Now the probability that an advanced level question will be answered correctly is:

Skill Level	Novice	Beginner	Intermediate	Advanced	Expert
Prob correct	*g*	*g*	*g*	*1-s*	*1-s*
Prob incorrect	*1-g*	*1-g*	*1-g*	*s*	*s*

Now the probability that an expert level question will be answered correctly is:

Skill Level	Novice	Beginner	Intermediate	Advanced	Expert
Prob correct	*g*	*g*	*g*	*g*	*1-s*
Prob incorrect	*1-g*	*1-g*	*1-g*	*1-g*	*s*

The key point is that we have reduced the number of probabilities required to the prior probabilities for the skill levels and just two other probabilities:

s—the probability that a user will slip up or not the answer to a question in a category that he should know the answer to,

g—the probability that a user will be lucky and get the right answer or do the right action in a category that he does not really have the knowledge for.

UPDATING TAKES PLACE IN LINEAR TIME

The Bayesian model for a single discrete probability variable X conditionally dependent on *n* different pieces of evidence is given by:

$$p(X = x_i \mid \vec{e}) = \alpha p(X = x_i) p(\vec{e} \mid X = x_i)$$

where $\vec{e}$

represents the evidence gathered and where α is a normalization constant. If we assume that each piece of evidence is independent of the others than this equation becomes

Now recall that each $p(e_j \mid X = x_i)$

will evaluate either to

(1-s) if the skill level x_i is the same or greater than the difficulty of the questions and the question is answered correctly, as would normally be expected.

s if the skill level x_i is the same or greater than the difficulty of the questions and the question is answered incorrectly, as would only be expected if a slip occurred.

g if the skill level x_i is less than the difficulty of the question and it is answered correctly, which would not normally be the case except for with a lucky guess.

(1-g) if the skill level x_i is less than the difficulty of the question and it is answered incorrectly, as expected as the question is categorized as too advanced for a user at this skill level.

For all question categories less than or equal to the current skill category (i.e., $e_j \leq x_i$), the student would normally answer all questions correctly or perform all actions correctly, barring slips. The probability of each correct answer is *(1-s)* and probability of each slip is *s*.

Similarly, for all question categories greater than the current skill category (i.e., $e_j > x_i$) the student would normally answer all questions correctly or perform all actions correctly, barring slips. The probability of each correct answer is *(1-s)* and probability of each slip is *s*.

All that is required to compute this formula is that we add up correct and incorrect answers in the various question categories and plug in these values along with the values for s, g, and the prior probability for $X = x_i$, the particular skill category being evaluated.

The equation on the next page shows the results.

e_i = number of questions answered correctly in category i

$\bar{e}_i$ = number of questions answered incorrectly in category i

$\sum_{i=1}^{i=j} e_i$ *is the number of questions that were answered correctly, as expected.*

$\sum_{i=1}^{i=j} \bar{e}_i$ *is the number of questions that were answered incorrectly, these are slips.*

$\sum_{i=j+1}^{i=n} e_i$ *is the number of questions that were answered correctly, these are lucky guesses.*

$\sum_{i=j+1}^{i=n} \bar{e}_i$ *is the number of questions that were answered incorrectly, as expected (too hard).*

$\therefore$ *the probability that the user has skill level x_i is $p(X=x_i|\vec{e})$ where*

$$p(X=x_i|\vec{e}) = \alpha\, p(X=x_i) \left((1-s)^{\sum_{i=1}^{i=j} e_i} \right) \left(s^{\sum_{i=1}^{i=j\_} e_i} \right) \left(g^{\sum_{i=j+1}^{i=n} e_i} \right) \left((1-g)^{\sum_{i=j+1}^{i=n\_} e_i} \right)$$

in the equation above α is a normalization constant easily computed by adding all the unnormalized values for each skill category. It is equal to the probability that the evidence vector occurs for any skill category. Formally,

$$\alpha = \sum_i p'(X = x_i \mid \vec{e})$$

where *p'* is computed using the formula above without the normalization constant α.

TRADE-OFFS WITH THIS APPROACH

Now we consider the pros and cons of this approach compared to Bayesian analysis without the simplifications we have made.

ADVANTAGES. First we consider advantages.

New performance items need only be classified by difficulty

The overhead in adding new performance items (e.g., multiple choice questions) is simple: they need only be placed in the proper question category. So one can easily add new questions to one category such as the beginner level, with no changes to the underlying algorithm. No new conditional probabilities must be ascertained from a subject matter expert.

Modularity: questions can be added or dropped

Similarly, particular questions can be dropped, or different numbers of questions in each category can be asked.

The model allows for both slips and lucky guesses

Unlike purely abductive approaches based on logic and searching for misconceptions, this approach is uncertainty based and explicitly allows for both slips and guesses in student performance.

Linear-time updating

Although updating belief networks is in general NP-hard, taking time exponential in the number of belief nodes, the algorithm presented is linear in the number of data items. It avoids the NP-hard problems by only dealing with one level of propagation, from data to the skill.

Simplified knowledge acquisition: only a small number of parameters

Typically only about a half dozen parameters are required: the probabilities for a slip and a lucky guess, plus the a priori probabilities of each skill level.

DISADVANTAGES. We have gained these advantages by trading off some complexities.

Propagation of values from one skill to another

First, only one skill can be modeled at a time. For more complex student models where we would like to model composite skills some means of propagating values upwards in the network is required. If Bayesian belief network updating algorithms are used this can take time that is exponential in the number of nodes.

Only binary-valued evidence data can be used

Second, only binary-valued evidence, that is either correct or incorrect is modeled with this approach. So we could not take into account answers that are correct but which took a long time to be answered and score those lower in some way than a correct answer given immediately.

OVERVIEW

The Stealth Application offers a way to build a user model, by using the system hooks to record the application features that a user utilizes. While the stealth application is running, it monitors the system hooks and keeps track of the buttons and menu items used. In general this means monitoring the WM_COMMAND messages and examining the HWND, LPARAM and WPARAM to see which window and command is being used. However some applications, particularly later versions of Microsoft Office and Microsoft Internet Explorer, do not send WM_COMMAND messages when Menu Items and Toolbar Buttons are selected. For those applications, we monitor mouse operations, if the mouse is clicked over a toolbar or menu item for these applications we query the application, through OLE Automation, to determine which button or menu item was selected.

STORES AMOUNT OF TIME MONITORED

The Stealth Application monitors when an application is started and ends, keeping track of the number of seconds that a monitored application has been used while being monitored.

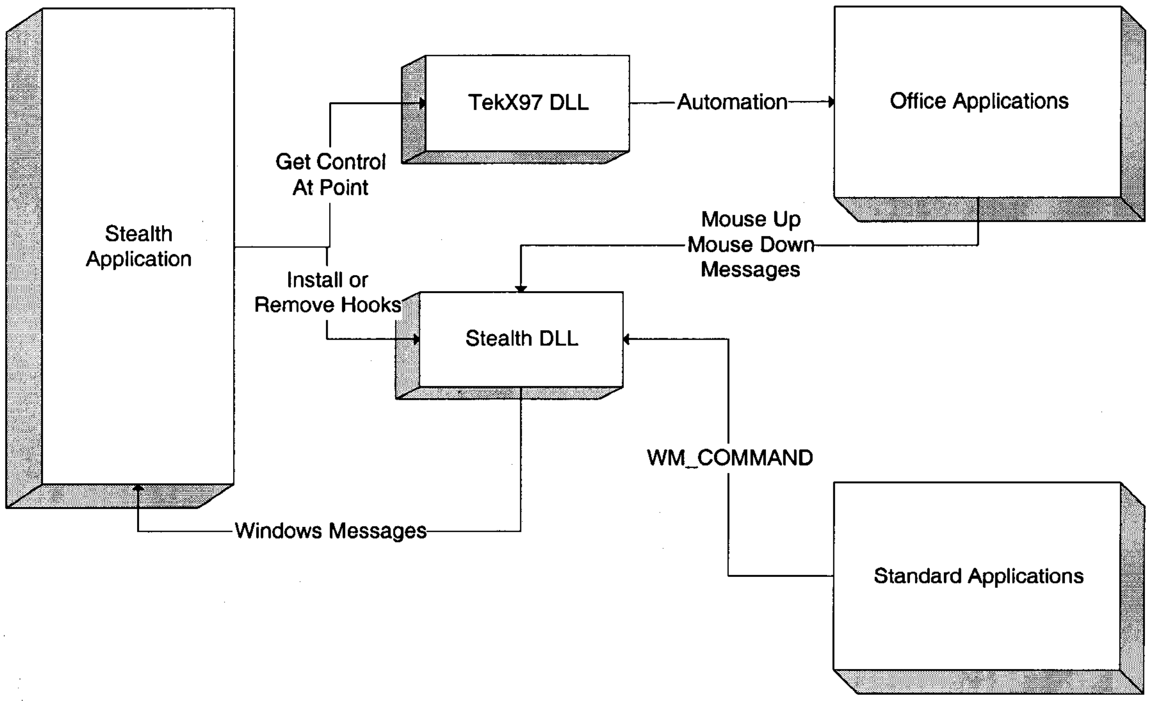
MODIFIES MENU

The Stealth Application modifies the menu for applications other than Microsoft Office Applications by grabbing their menu and adding items to it. It monitors the system hooks and can react to these messages being selected.

Figure III-1.

Describes the basics for how the Stealth Application

Works. The Stealth DLL is the center for the entire application. It is where the callback functions reside.



COMPONENTS

Component	Description
Stealth Application	The Main Monitoring Application
Stealth DLL	The location where the Callback Functions are located.
TekX97 DLL	Automation connection for Office Applications
Office Applications	Microsoft Office Applications
Standard Applications	Applications, which use WM_COMMAND when menu items or toolbar buttons are selected.

STEALTH DLL

The Stealth DLL is where all of the callback functions reside. It gets notified whenever a mouse up, mouse down, and Messages before they are sent to the Windows Message Queue (only WM_COMMAND, WM_CREATE and WM_DESTROY are looked processed from these messages.)
Use the InstallHook function to install and uninstall the hooks.

TEKX97 DLL

The TekX97 DLL is used to communicate to the office applications. The TekX97 DLL uses OLE automation to communicate with the MS Office applications.

MOUSE UP AND MOUSE DOWN

Stealth DLL is notified that a Mouse-Down Operation has occurred.
Posts a message to the Stealth Application informing it of the Location and HWND.
Stealth Application Checks to see if this is an Office Command Bar (Menu or Toolbar). If not then processing of this message ends.
Records the location and window that the mouse operation occurred.

Stealth DLL is notified that a Mouse-Up Operation has occurred.
Posts a message to the Stealth Application informing it of the Location and HWND.
Stealth Application Checks to see if this is an Office Command Bar, if not then processing of this message ends.
Using TekX97.DLL Gets the control that the Mouse-Down Operation occurred on.
Using TekX97.DLL Gets the control that the Mouse-Up Operation occurred on.
If Mouse Down and Mouse Up controls match, then records a hit to that control.

WM_COMMAND

Stealth DLL is notified that a WM_COMMAND message is going to occur.
Posts a message to the Stealth Application.
Stealth Application checks to see if this message came from a monitored application. If not then processing of this message ends.
Looks at the WPARAM and determines which control was used and records this event.

WM_CREATE

Stealth DLL is notified that a WM_CREATE message is going to occur.
Posts a message to the Stealth Application.
Stealth Application checks to see if this message came from a monitored application. If not then processing of this message ends.
Records a start time for this application and modifies its HWND.

WM_DESTROY

Stealth DLL is notified that a WM_DESTROY message is going to occur.
Posts a message to the Stealth Application.
Stealth Application checks to see if this message came from a monitored application. If not then processing of this message ends.
Records an End time for this application. If this is the last application then updates the database.

COPYRIGHTS AND TRADEMARKS

The table below shows products with copyrights and trademarks that are referred to in this report. The column on the right shows the shorter term used to refer to it.

Copyrighted Product	*Name(s) used to refer to it in this report*
Netscape Communicator 4.01	Netscape
Visio Technical 4.5 for Microsoft Windows	Visio
Microsoft™ Word 97	Word
Microsoft™ Word 6 for Windows 95	Word 6/95
Microsoft™ Excel 97	Excel
Microsoft™ PowerPoint™ 97	PowerPoint
Microsoft™ Access 97	Access
Microsoft Agent	ActiveX agent, the Genie
Microsoft Internet Explorer 3.02	Internet Explorer